National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

UNIVERSITY OF ALBERTA

Obvious Abduction

by

© Dekang Lin

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Doctor of Philosophy

Department of Computer Science

Edmonton, Alberta
Spring 1992

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-73020-X

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Dekang Lin

TITLE OF THESIS: Obvious Abduction

DEGREE: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: 1992

Permission is hereby granted to the University of Alberta library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's written permission.

(Signed) Dekang Lin

Dekang Lin
#320, 77 University Crescent
Winnipeg, Manitoba, Canada, R3T 3N8

Date: Mar 23, 1992

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **Obvious Abduction** submitted by **Dekang Lin** in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

R. Goebel
R. Goebel, Dept. of Computer Science
Superviser

J. Pelletier, Dept. of Computer Science

J. Hoover, Dept. of Computer Science

Wm W. Armstrong
W.W. Armstrong, Dept. of Computer Science

A.N. Al Hussaini
Dept. of Statistics and Applied Probability

E. Charniak
Dept. of Computer Science, Brown University

Date: Mar 19, 1992

To my parents

Abstract

Abduction, or the inference to the best explanation, is a pervasive phenomenon, both in science and in everyday life. Recently, there is a growing awareness that key tasks in many different areas of AI can be cast as abduction. This research is motivated by an apparent paradox. On the one hand, most computation models of abduction have proven to be intractable. On the other hand, humans are capable of making certain kinds of abductive inferences in a flash. To explain this paradox, we propose a theory of abduction that purports to cover the kinds of abductive inferences humans make efficiently. We use the term *obvious abduction* to loosely refer to these kinds of abductive inferences.

The main contributions are as follows:

Generality: The theory we propose is applicable across several application domains such as diagnosis, plan recognition, and natural language parsing. Such a unified theory will not only facilitate more accurate characterization and understanding of abductive reasoning, but also foster cross fertilization among different applications of abductive reasoning.

Efficiency: The complexity of our abduction algorithm is polynomial in the size of the knowledge base, and exponential in the number of observations to be explained. Therefore, abduction is relatively efficient when the number of observations is small.

Probabilistic Justification: The knowledge representation scheme in obvious abduction allows probabilistic/statical knowledge to be represented. The inference algorithm is able to compute the probability of explanations and the most probable explanation is preferred.

Acknowledgments

I am most indebted to my supervisor, Randy Goebel, who contributed guidance and insight, and provided a wonderful research environment. His insightful questions, helpful suggestions, and constructive criticisms played an essential role during the course of this research.

I am also indebted to my committee members A.N. Al Hussaini, W.W. Armstrong, J. Hoover and J. Pelletier for their valuable comments on the draft of the dissertation.

Special thanks to my external examiner Eugene Charniak, who provided detailed comments and suggestions.

The weekly research meeting with "Goebelians" had always been very helpful. Thanks are due to Robert Baron, Andrew Edgar, Aditya Ghose, Abdul Sattar, Scott Goodwin and Charles Truscott. Scott deserves special thank for always being available whenever I needed his inputs.

I would like to thank Fred Popowich for his insightful directions on parsing, Peter van Beek for his comments and suggestions on plan recognition.

I am grateful to Judea Pearl, who took the time to point out the weaknesses in an earlier paper and suggest the approach of characterizing causation events in terms of lower level micro events.

Thanks to Yong Hu and Haiying Wang who kept lunch "meetings" entertaining over the years.

Finally, my deepest gratitude to Yihua Lou Yuan, who is the source of constant support and encouragement, whose love made this endeavor worthwhile. I wish I could do as much to see her through PhD.

List of Figures

1 A causal network for auto diagnosis 17
2 A causal scenario 18
3 The plan hierarchy of a cooking world 21
4 A scenario 21
5 An example CFG grammar 23
6 Translation of CFG rules into a network 23
7 A grammar network 24
8 The input sentence 24
9 A local tree 25
10 A generalized subtree 26

11 More than 50% of dogs bark 30
12 **Lp** augments FOL conceptualization with a probability space . . 31
13 `rename-file` has two steps 32
14 Conceptualization of the world 33
15 The BNF syntax of $\mathbf{S}^3$ 39
16 The CF-Diagram for the cooking world 42
17 Path Preemption 42
18 Local tree 43
19 A walk tree 44
20 A scenario of `make-spaghetti-marinara` 45
21 Decomposition of a scenario 48

22 The operations of `Node` class 56
23 The `add` operation of `ItemStore` class 57
24 A trace of an item 61
25 The decomposition of a trace 63

26 Specificity and causal independence 74
27 Misfiring scenarios 76
28 Specificity of explanations 79
29 Sorting a list 80
30 A `quickSort` algorithm 83
31 An insertion sort algorithm 84
32 An alternative for `partition1` 84

33 An alternative for `partition11` 85
34 A CF-Diagram of sorting algorithms 86
35 Observations about a sorting program 88
36 The explanation of actions in Figure 35 88

37 Theorist reasoning architecture 98
38 Hunt/Rob Hierarchy 103
39 Parsimonious covering theory 105
40 Parsimonious covering theory 106
41 Set covering model of abduction 106
42 A working/shopping world 109
43 "John went to the supermarket. He put on the uniform" 111
44 Buy milk scenario 112
45 Bureaucrat in city hall 112
46 Representation of domain knowledge in abduction 113
47 Isa link must be treated differently 116
48 Representing multi-cause interaction in obvious abduction 117
49 Definitions of explanation 119
50 Explanation ranking in abduction 121
51 Most probable explanation may not be minimal 122

Contents

1 **Introduction** **1**

1.1 Motivation . . . 1

1.2 Characterizing abduction . . . 4

1.2.1 What is an explanation? . . . 5

1.2.2 What is a good explanation? . . . 6

1.3 Objectives . . . 8

1.3.1 A unified theory . . . 8

1.3.2 Efficiency of abductive inference . . . 9

1.3.3 Principled evaluation of explanations . . . 9

1.4 Approach . . . 10

1.5 Limitations . . . 12

1.6 Overview of the chapters . . . 13

2 **Applications of obvious abduction: an informal preview** **15**

2.1 Diagnosis . . . 15

2.2 Plan recognition . . . 18

2.3 Parsing . . . 22

2.3.1 Context-free grammar . . . 22

2.3.2 A network representation of CFG . . . 23

2.4 Summary . . . 25

3 **Obvious abduction: a formal theory** **27**

3.1 Logic and probability 27

  3.1.1 Logic 28

  3.1.2 Probability 28

3.2 The Language **Lp** 29

3.3 Object-Feature-Attribute Representation 31

  3.3.1 Conceptualization 31

  3.3.2 Axiomatization 33

3.4 The language $\mathbf{S}^3$ 37

3.5 Scenarios 41

3.6 Abductive reasoning problem 45

3.7 Probabilities of explanations 47

  3.7.1 A special case 50

4 A generic message passing algorithm **52**

4.1 Overall description 52

4.2 Object-oriented implementation 54

  4.2.1 An example 58

  4.2.2 Trace of an item 60

4.3 Correctness of the algorithm 62

4.4 Complexity of the algorithm 66

5 Applications of obvious abduction: revisited **69**

5.1 Diagnosis 69

  5.1.1 Causality 69

  5.1.2 Logic interpretation of causal network 71

  5.1.3 Causal independence 72

  5.1.4 Examples from auto diagnosis 75

5.2 Plan Recognition 77

  5.2.1 Specificity 77

5.2.2 Recursive plans 79
5.3 Parsing as abduction 91
5.3.1 Translation of CFG into $\mathbf{S}^3$ 91
5.3.2 Complexity analysis of CFG parsing 93
5.3.3 ID/LP parsing 94
5.3.4 Use of attributes 95

6 Related work **97**
6.1 Domain independent abduction 97
6.1.1 Theorist 97
6.1.2 Levesque 99
6.1.3 TACITUS 101
6.1.4 Ng & Mooney 101
6.1.5 Thagard 102
6.1.6 Kautz 103
6.1.7 PCT 104
6.1.8 Bylander *et al.* 105
6.1.9 Bayesian Networks 107
6.2 Comparison of abduction theories 112
6.2.1 Knowledge representation 113
6.2.2 Problem formulation 118
6.2.3 Explanation ranking 120
6.2.4 Computational complexity 123
6.3 Spreading activation in semantic networks 124

7 Conclusions **126**
7.1 Summary of Contributions 126
7.2 Future Work 128
7.2.1 The general framework 128

7.2.2 Diagnosis 129
7.2.3 Plan recognition 129
7.2.4 Natural language parsing 130

List of Figures

1 A causal network for auto diagnosis 17
2 A causal scenario 18
3 The plan hierarchy of a cooking world 21
4 A scenario 21
5 An example CFG grammar 23
6 Translation of CFG rules into a network 23
7 A grammar network 24
8 The input sentence 24
9 A local tree 25
10 A generalized subtree 26

11 More than 50% of dogs bark 30
12 **Lp** augments FOL conceptualization with a probability space . . 31
13 `rename-file` has two steps 32
14 Conceptualization of the world 33
15 The BNF syntax of $\mathbf{S}^3$ 39
16 The CF-Diagram for the cooking world 42
17 Path Preemption 42
18 Local tree 43
19 A walk tree 44
20 A scenario of `make-spaghetti-marinara` 45
21 Decomposition of a scenario 48

22 The operations of `Node` class 56
23 The `add` operation of `ItemStore` class 57
24 A trace of an item 61
25 The decomposition of a trace 63

26 Specificity and causal independence 74
27 Misfiring scenarios 76
28 Specificity of explanations 79
29 Sorting a list 80
30 A `quickSort` algorithm 83
31 An insertion sort algorithm 84
32 An alternative for `partition1` 84
33 An alternative for `partition11` 85
34 A CF-Diagram of sorting algorithms 86
35 Observations about a sorting program 88
36 The explanation of actions in Figure 35 88

37 Theorist reasoning architecture 98
38 Hunt/Rob Hierarchy 103
39 Parsimonious covering theory 105
40 Parsimonious covering theory 106
41 Set covering model of abduction 106
42 A working/shopping world 109
43 "John went to the supermarket. He put on the uniform" 111
44 Buy milk scenario 112
45 Bureaucrat in city hall 112
46 Representation of domain knowledge in abduction 113
47 Isa link must be treated differently 116
48 Representing multi-cause interaction in obvious abduction 117

49 Definitions of explanation 119

50 Explanation ranking in abduction 121

51 Most probable explanation may not be minimal 122

List of Tables

5.1 Items in the scenario in Figure 36 89

6.2 Complexities of abduction theories 123

Chapter 1

Introduction

1.1. Motivation

Abduction is the inference to the best explanation. It is a kind of non-deductive inference that uses the following pattern: let α and β be two sentences,

$$\begin{array}{ll} \text{Given} & \alpha \rightarrow \beta \\ \text{and} & \beta \\ \hline \text{Infer} & \alpha, \text{ because, if } \alpha \text{ is true, } \beta \text{ would be explained.} \end{array}$$

The word "abduction" was coined by the philosopher Charles Peirce (1839–1914) to provide an account of the process of discovery:

> The first stating of a hypothesis and the entertaining of it, whether as a simple interrogation or with any degree of confidence, is an inferential step which I propose to call *abduction*. ... I call such inference by the peculiar name, abduction, because its legitimacy depends upon altogether different principles from those of other kinds of inference [49, pp.236–7].

Abduction, together with deduction and induction, are the three basic forms of inference. While deduction and induction are widely known in AI, abduction had received relatively less attention until recently and is the least understood of the three.

Yet abduction is a pervasive phenomenon, both in science and in everyday life [75]. The discovery of scientific theories is first motivated by some puzzling phenomena. Scientists then generate hypotheses that would explain the observations.

For example, in 1821, it was observed that the aberrations in Uranus's orbit was incompatible with the positions derived from Bouvard's tables. Leverrier found that this tension could be resolved by postulating the existence of some mass having the property to generate Uranus's observed positions. He hypothesized, abductively, that there was an as yet unobserved planet beyond Uranus which was perturbing its motion, because this would explain the discrepancy between Uranus's actual and predicted orbit. There were also other explanations of the observation. For example, some astronomers speculated that Newton's theory of gravitation did not hold at great distances. Leverrier's explanation was later confirmed by the discovery of Neptune, which also refuted doubt on the applicability of Newton's theory universal gravitation.

In our everyday life, we constantly form hypotheses to explain the behavior of other people, filling in the gaps around what we know. Suppose we read the following simple story [8, p.555]

> Fred went to the supermarket. He found the milk on the shelf. He gave the checkout clerk some money and left.

Since a scenario of Fred buying milk in a supermarket involves all the observed actions, we infer naturally that Fred bought the milk. But note that the story never actually said this. It didn't even say Fred took the milk from the shelf. We made the inference by abduction.

For another example, suppose Fred found his mailbox empty, he might infer that his wife was home, because this would explain the mailbox being empty. This inference is not deductive because there are other explanations as well. For instance, Fred did not have any mail that day, or there was a postal strike, etc. If Fred further noticed that the living room light was on, the abductive conclusion would be reinforced.

There is a growing awareness that key tasks in many different areas of AI can be cast as abduction or inference to the best explanation.

- In expert systems, diagnosis is clearly an example of abductive reasoning. In medical diagnosis, a patient's case involves a set of observations, a diagnosis may be reached when one or more disorders have been found to be capable of causing all the observations about the patient. In device diagnosis, the diagnostician may start with a set of observations about the abnormal behaviours of the device and try to find out how such behaviours can be accounted for by the failure of certain components of the device.

- Applications of abductive reasoning have also been found in intelligent interface research. In order to provide cooperative response to requests to systems such as as expert systems [52], databases [33, 20, 6], these systems must be able to infer their user's underlying goals in using the system from previous requests. Such inference is abductive in the sense that its conclusion explains why the user has asked these particular questions.

- Many tasks of natural language understanding have been classified as abductive reasoning. The understanding of extended discourse involving the actions of people, such as newspaper articles or stories, is a problem which requires one to reason abductively to generate an explanation that pieces together the observed actions [8]. In word sense disambiguity, the solution is an explanation of how the choice of the word senses in a sentence fit together.

- Visual understanding can be viewed as a problem of finding an explanation for the array of light intensities, e.g., as picked up by a TV camera. For example, in the domain of interpreting sketch maps, the MAPSEE system [26] is given a bit map image, a collection of feature detection procedures (e.g., edges, lines, regions), and a database of knowledge that describes potentially present scene objects (e.g., roads, rivers, bridges). The system seeks a consistent set of scene labels to image domain features, which, together with general knowledge about scene and image domain knowledge, accounts for the observations—the actual image under scrutiny.

Early research on abductive reasoning in AI was largely conducted in isolated areas such as abductive diagnosis (e.g., CASNET [80], INTERNIST [43]), associative reasoning in semantic networks [59, 9], story understanding [67] and discourse analysis [1]. Although they have achieved significant results, the methods used for abductive reasoning are mostly *ad hoc* and difficult to generalize.

To remedy the deficiencies of domain-dependent heuristic systems, researchers began to investigate domain-independent formal theories of abductive reasoning. The earliest attempts include Theorist [56], Surface deduction [12], Set Covering Model [61] (later evolved into Generalized Set Covering Theory [50]), Maximum a-posteriori (MAP) Assignment [47, 69], and Kautz's formal theory of plan recognition [34]. The domain-independent power of these theories or systems largely results from a domain-independent knowledge representation. The formal formulations of the abductive reasoning problem and declarative specifications of its solution allow easier analysis and comparison of their properties and behaviour.

Here we present a domain-independent formal theory of abduction. This research is motivated by an apparent paradox: On the one hand, most computation models of abduction have been proven to be inherently difficult. On the other hand, humans are capable of making some abductive inferences in a flash. To explain this paradox, we propose a theory of abduction that purports to cover the kinds of abductive inferences humans make efficiently. We use the term *obvious abduction* to loosely refer to these kinds of abductive inferences.

1.2. Characterizing abduction

Although the typical description of abduction as "inference to the best explanation" is intuitively meaningful, it is nonetheless too underspecified to be used in a computational theory of abduction. A satisfactory definition of abduction must encompass at least following:

1. It must answer the question "What is an explanation?" by elucidating the linkage between the *explanans* and the *explanandum*.
2. It must answer the question "What is a good explanation?" by clearly specifying the conditions for the correctness and the relative strength of explanations.

1.2.1. What is an explanation?

Generally speaking, the purpose of an explanation is to render facts intelligible to a mind seeking to understand. The correct explication of the notion of explanation is still a topic of discussion and dispute in philosophy. Computational theories of abduction usually proceed with some operational definitions of explanation. Such operational definitions are satisfactory as long as they match the intuitive notion of explanation in the domain (or domains) being considered.

The following have been proposed as necessary conditions for an explanation:

Consistency: An explanation must be consistent with background knowledge. The necessity of consistency is self-evident wherever consistency can be defined, because inconsistent explanations cannot possibly be true.

Causality: The relationships between hypotheses and the observations they explain are not simply logical implications. For example, "Mark is 3 years old" logically implies "Mark is 3 or 4 years old," however, it does not explain the latter. Some researchers have argued that the relationships between hypotheses and observations must be causal [15, 32]. Others believe there are cases where the relationship is simply associative [82].

Relevance: The quality of an explanation depends not only on the observation but also on the goal of the abductive reasoner: that is, how the abductive conclusions are to be used. An explanation must be relevant to the goal and as informative as necessary, but not more so.

For example, suppose on a cold winter day, your colleague was late for work. You might explain this by hypothesizing that their car was unable to start due to the cold weather. If you were not going to fix the car, a further hypothesis that the car was difficult to start because of a problem with the automatic choke is irrelevant.

1.2.2. What is a good explanation?

Abduction is unsound inference. Without some evidence that the explanation is the best, abduction can easily lead to non-sense. The following story from [75, p.75] examplifies this point:

> Recall the grisly joke about the scientist who cut off one of a frog's legs and said, "Jump", and the frog jumped. Then he cut off two more of its legs and said, "Jump", and the frog more or less jumped. Finally, he cut off the remaining leg and said, "Jump", but the frog did not jump. The scientist accordingly abduced that frogs with no legs are deaf.

An abductive reasoner must find the best one or best ones, according to some criteria. There is no consensus among AI researchers or philosophers about the criteria of ranking explanations. Some believe that such a problem is just a matter of pragmatics and that no logical criterion can be used to support the choice [36, 62].

Researchers have identified the following factors as contributing to the quality of an explanation.

Simplicity: *Occam's razor,* which postulates that, everything else being equal, a simpler explanation is preferred, is the most well-known and often-used criterion for ranking explanations [50, 34, 36]. The idea of simplicity has been used quite uncritically, as though it were obvious what simplicity is, and why it should be valuable.

Coherence: The parts that constitute an explanation must fit together with each other and show the relatedness of the observations [76, 44]. For example, given a sentence "John was happy. The exam was easy," the explanation (interpretation) that *John did well on the easy exam* is preferable to the explanation *John is an optimist.* This is because the latter does not show the relatedness between "John was happy" and "The exam was easy" [44].

Certainty: The certainty of an explanation depends both on the certainty of hypotheses it involves, as well as the arguments connecting them to the observations. Probability theory is a branch of mathematics that specifically deals with the certainty of information and therefore is apparently a rational measurement of the certainty of the explanations. How to use probability in this context, however, is not a simple issue.

Specificity: It is generally accepted that the specificity level of an explanation has something to do with its quality. However, how specificity affects the quality of explanation is a subject of debate. Proposals on the issue include: most specific abduction [12, 54], least specific abduction [72], chain specific abduction [73].

Consilience: One explanation is said to be more consilient than another if it explains more classes of facts than the other does [75]. For example, the general theory of relativity proved to be more consilient than Newtonian mechanics by explaining the perihelion of Mercury and the bending of light in a gravitational field.

Alternative explanations: The previous criteria are concerned only with the properties of a single explanation. The quality of an explanation does not only rely on its own properties, but also on their ranking with respect to alternative explanations. When all the other alternative explanations have been deductively eliminated, the abductive inference falls into the deductive inference pattern of "disjunctive syllogism" [32]:

P or Q or R or S or ...
But not-Q, not-R, not-S, ...

Therefore, P.

The abductive reasoning systems and theories often use a heuristic interpretation of one of the criteria. For example, Peng and Reggia [61] and Levesque [36] used Occam's Razor to rank explanations, where simplicity is interpreted as minimality of the set of hypotheses used to explain the observations. Ng and Mooney [44], on the other hand, claimed that Occam's Razor is not sharp enough. So they invented a heuristic coherence measure instead. Since none of the proposed criteria is dominant, relying on one of them to rank explanations is bound to produce undesirable results under certain circumstances. Moreover, the lack of a formal basis for heuristic interpretations of simplicity and coherence is also unsatisfactory.

1.3. Objectives

The three main objectives of this research (which are also the contributions of this dissertation) are:

1.3.1. A unified theory

We propose a unified theory of abductive reasoning which is applicable to several applications domains, such as diagnosis, plan recognition, and natural language parsing. Such a unified theory will not only facilitate more accurate characterization and understanding of abductive reasoning, but also foster cross fertilization among different applications of abductive reasoning. For example, the use of probability has been extensively studied in diagnosis. Here the same technique is transferred to plan recognition and parsing to resolve ambiguities. On the other hand, probabilistic approaches to abductive diagnosis [50, 39] that are weak in consistency checking are augmented with feature and attribute value constraints that are commonly used in parsing and plan recognition.

1.3.2. Efficiency of abductive inference

In general, abductive reasoning is inherently difficult. Even the abductive reasoning systems with modest expressive power have been proved to be NP-complete [5]. On the other hand, humans are able to make certain kinds of abductive inference with ease. We note that, in these cases, the observations to be explained are explicit and the number of observations to be explained is relatively small.[1] This suggests that, by taking into consideration such restrictions on the input, it is possible to have a relatively efficient abductive reasoning procedure. It is no coincidence, therefore, that in the NP-completeness proofs of various abduction schemes [5], the number of observations is assumed to be unbounded.

The complexity of our abductive inference algorithm is polynomial in the size of the knowledge base and is exponential in the number of observations to be explained when applied to some domains.

1.3.3. Principled evaluation of explanations

One of the most important issues in abduction is the ranking of explanations. Most previous approaches to abduction use heuristic interpretation of simplicity [36, 34, 50] or coherence [44, 76] to rank the explanations. The explanations are usually used by the reasoning agent to make predictions. With heuristic measurements, it is hard to compute the utility of explanations. Most utility theories are based on probability measurement. We define explanations to be scenarios, which are descriptions of a set of related objects. The definition of a scenario incorporates the requirement for consistency, coherence, and relevance. The best explanation is the most probable one among the maximally specific scenarios that, together with the background knowledge, entail the observations.

---

[1] In medical diagnosis, even though the data collected by a physician may be large, many of them, such as age and gender, needs no explanation.

Summarizing, the theory of abductive reasoning proposed here is general enough to be applied in several disparate domains; is efficient when the observations are explicit and small in number; and uses formal probability-based criteria to rank the explanations.

1.4. Approach

In the approach we have taken, domain knowledge is organized around objects. The objects are the individuals in the universe of discourse that are of primary interest, such as instances of disorders and symptoms in diagnosis, plans and actions in plan recognition, and instances of grammatical categories in parsing. An object is characterized by a set of attributes. The structural relationships between objects are represented by features, which are unary functions mapping an object to some other objects. Objects are grouped into categories. The categories are organized in generalization-specialization hierarchies.

The objects, features and attributes are axiomatized in the language **Lp**, an extension of the first order logic developed by Bacchus [2] to represent and reason with both logical and probabilistic information.

An observation is a description of an object. An explanation of a given set of observations is a scenario that logically entails the observations and is consistent with the knowledge base. A scenario is a description of a set of related objects. The scenarios are defined in such a way that more specific information supersedes less specific information. The probability of a scenario is a probability term in **Lp** and can be computed from statistical information about the domain, which is also represented in **Lp**. The abductive reasoning problem is that of finding the most probable scenario that explains the observations.

Since we have used only a restricted subset of **Lp** to represent the observations, scenarios and domain knowledge, a specialized inference algorithm can be used to find the most probable explanation. The axioms representing the domain knowledge are translated into a network whose nodes represent categories and whose links represent features. Abductive inference is performed by a message passing algorithm operating over the network. Each of the nodes in the network is also a computing agent that communicates with neighbouring agents by passing messages across the links in the network. The messages represent partial explanations of the observations, which are explanations of a subset of observations. The partial explanations are combined with others at the nodes to construct explanations for larger subsets of observations. Dynamic programming techniques are employed so that the most probable scenarios can be found without enumerating all those possible.

A major source of intractability for many abductive reasoning systems is consistency checking. If the representation language is standard form of first order logic,[2] consistency checking is a semidecidable process, thus making the overall abduction undecidable. In our representation language, we restrict the consistency conditions to local constraints and attribute percolation constraints, so that the consistency of a scenario can be established locally and efficiently by the computing agents.

We demonstrate the generality of our representation language and generic abductive reasoning algorithm by applying them in diagnosis, plan recognition and parsing. Previous algorithms for abductive diagnosis, plan recognition and parsing are apparently quite distinct. [3] Here we show that they share the same representation language and the same generic message passing algorithm. They differ only in the structure and contents of the messages and the local constraints on the propagation and combination of messages at each node in the network.

---

[2] Some subsets of first order logic are decidable, e.g., unary first order logic (cf. Hilbert)

[3] Except Vilain's proposal [79] to use parsing algorithms in plan recognition and Bayesian network based apporach to diagnosis and plan recognition [7, 68]

1.5. Limitations

Any theory has its limitations. Obvious abduction is no exception. The major limitations of obvious abduction is listed as follows:

Observations are explicit: We assume that the observations are explicitly given to the abductive reasoner. In the application domains we are considering, this is a reasonable assumption.

Observations are small in number: We also assume the number of observations is small. This restriction is important to the computational efficiency of obvious abduction. The complexity of our abductive inference algorithm is exponential in the number of observations in some domains. We note that humans also have difficulty dealing a large number of observations that need explanation. For example, a control room operator can easily be overwhelmed by a large number of simultaneous alarms [64].

Explanations involve no new concepts: The explanations inferred from obvious abduction do not involve relations or functions that are unknown to the knowledge base. For example, in diagnosis, the types of events and the causal relationships among the events in an explanation all belong to the set of event types and the set of causal relations in the knowledge base.

Restriction in knowledge representation: The applicablity of obvious abduction is also limited by its representation language. The details of the knowledge representation language can be found in Chapter 3. Roughly speaking, the knowledge must be represented in the patterns of the form $A \rightarrow B$. In diagnosis, this restrictions implies that the multiple cause interaction must be modeled by a noisy-or gate [47].

These limitations make the meaning of "obvious" in Obvious Abduction more clear. That is, obvious abduction is a kind of abduction where

1. observations are explicit and small in number;
2. explanations do not involve novel concepts;
3. domain knowledge can be represented in the language defined in Chapter 3.

Therefore, obvious abduction is inadequate as a model of detective's inferencing where identifying which observations need explanation is no easy task, or as a model of scientific discovery where explanations often involve novel concepts. On the other hand, abductive inferences in these domains are far from being obvious to most humans.

1.6. Overview of the chapters

The remaining chapters of this dissertation is organized as follows: The next chapter is an informal preview of the applications of obvious abduction, including diagnosis, plan recognition and parsing.

Chapter 3 and Chapter 4 will present a formal theory of obvious abduction in detail. Chapter 3 is concerned with the knowledge representation and problem formulation for obvious abduction. A category-feature-attribute-based representation language is presented there. Chapter 4 contains a generic message passing algorithm along with an argument for its correctness and complexity analysis.

Chapter 5 shows how the theory and algorithm can be applied in the diagnosis, plan recognition and parsing problems. The chapter also contains the comparisons of our approach to these problems with others' and the advantages offered by obvious abduction.

Chapter 6 summarizing our comparison of obvious abduction with other general purpose abduction theories and systems. Chapter 7 concludes by summarizing the overall contributions, and suggesting further research.

Chapter 2

Applications of obvious abduction: an informal preview

In this chapter we present the intuitive ideas of obvious abduction and show how it applies to the problem of diagnosis, plan recognition and parsing. Each section will introduce of the problem domain and show how the problems can be cast as an abductive reasoning problem. By means of examples, we show that the knowledge used to solve these problems can be represented by a network, and that solving these problems amounts to finding a connection in the network between the observations. We will formalize these ideas in later chapters. Here, we hope to present an overall picture of what we are attempting.

2.1. Diagnosis

Diagnosis was one of the earliest applications of abduction in AI [57]. It is typically defined as the problem of inferring faults or disorders from symptoms or manifestations. As with many other abductive diagnosis systems and theories, causality is used here as a pivotal mechanism. The underlying motivation for developing causal models in diagnosis is the conviction that such a model may let the computer understand its domain more profoundly and therefore reason about the domain in a more general fashion.

In the proposed abductive diagnosis theory, the domain knowledge is represented by a network. The nodes in the causal network represent events, which are the presence of symptoms, disorders or intermediate states. There are three types of links in the causal network: causal links, "isa" links and specialization links. The semantics of causal links will be formally defined in Chapter 5. Here, it is sufficient to understand that they represent causal relationships between events. The generalization and specialization relationships are represented by "isa" and specialization links

Figure 1 is a causal network for automotive engine diagnosis, where solid arrows represent causal links and each of the bidirectional gray arrows represents a pair of "isa" and specialization links. The upward direction represents the "isa" link and the downward direction represents the specialization link.

If one event is capable of causing another event, then there is a causal link from the former to the latter. For example, since `oil-pressure-too-low` may cause `overheating` of engine, there is a causal link from the node `oil-pressure-too-low` to the node `overheating`.

If an event is a subclass of another event, then there is an "isa" link from the former to the latter and a specialization link from the latter to the former. For example, `oil-fouled-spark-plug` is a kind of `fouled-spark-plug`. Therefore there is a pair of "isa" and specialization link between them.

The "isa" links in the causal network allow inheritance of causal properties. For example, `oil-fouled-spark-plug` inherits the ability of causing `bad-spark` from `fouled-spark-plug`. The specialization links allow inheritance of causal explanations. For example, the explanation of `improper-idle-speed` from `bad--carburetor-chip` may be inherited by `idle-speed-too-slow`. In other words, since `idle-speed-too-slow` is a subclass of `improper-idle-speed` and `bad--carburetor-chip` explains `improper-idle-speed`, `bad-carburetor-chip` also

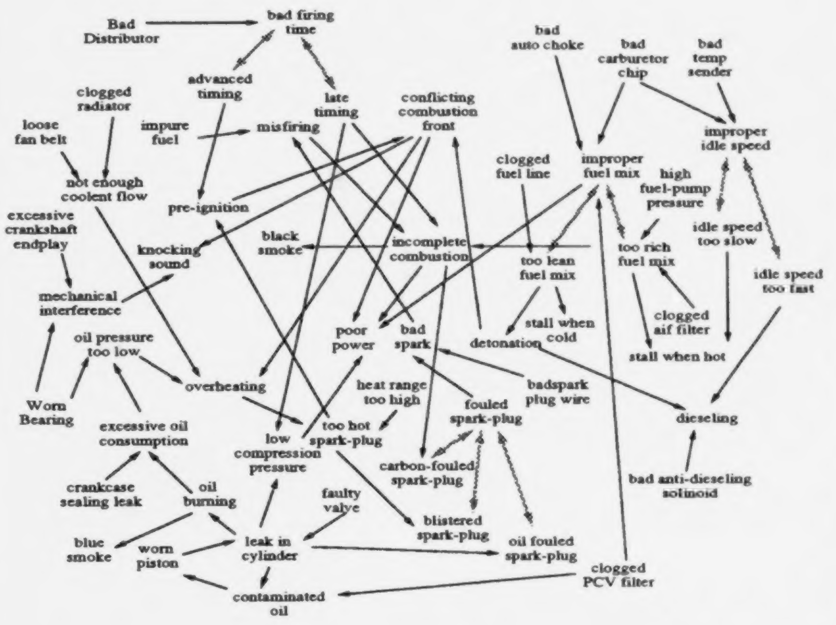

Bad Distributor
bad firing time
advanced timing
late timing
clogged radiator
loose fan belt
impure fuel
misfiring
conflicting combustion front
not enough coolent flow
pre-ignition
excessive crankshaft endplay
knocking sound
black smoke
incomplete combustion
mechanical interference
oil pressure too low
poor power
bad spark
detonation
overheating
heat range too high
Worn Bearing
excessive oil consumption
too hot spark-plug
low compression pressure
fouled spark-plug
carbon-fouled spark-plug
crankcase sealing leak
oil burning
faulty valve
blistered spark-plug
blue smoke
worn piston
leak in cylinder
oil fouled spark-plug
contaminated oil
bad auto choke
bad carburetor chip
bad temp sender
improper idle speed
clogged fuel line
improper fuel mix
high fuel-pump pressure
too lean fuel mix
too rich fuel mix
idle speed too slow
idle speed too fast
stall when cold
clogged aif filter
stall when hot
badspark plug wire
dieseling
bad anti-dieseling solinoid
clogged PCV filter

Figure 1: A causal network for auto diagnosis

explains `idle-speed-too-slow`.

One of the characteristics of causality is that an effect can also be a cause in a chain of causations. For example, `loose-fan-belt` causes `not-enough--coolant-flow`, which, in turn, causes `overheating` of engine. Such a causal chain forms a path in the causal network.

A causal path specifies how an event may lead to another event through a chain of causations. A subtree of the causal network specifies how an event may lead to multiple other events through chains of causations. We say such a subtree is a **scenario**, which is a generalization of causal path. Such a scenario represents a composite hypothesis, which contains combinations of symptoms, disorders and intermediate states as well as the interactions among them, and can be used as explanations of a set of observations.

For example, Figure 2 shows a scenario, where `too-rich-fuel-mix` caused `stall-when-hot` and `incomplete-combustion`, which, in turn, caused `carbon-fouled-spark-plug` and `black-smoke`.

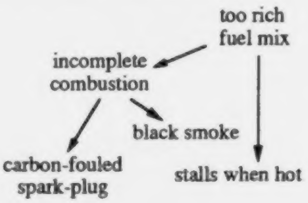


Figure 2: A causal scenario

An **observation** is represented by a node in the causal network. Given a set of observations, the task of the diagnostic system is to find explanations of the observations. An **explanation** of a set of observations is a sub-tree of the causal network that connects all the observation nodes. Such an explanation can be regarded as a tentative reconstruction of the causal evolution which has led to the observations.

For example, suppose `stall-when-hot`, `black-smoke`, and `carbon-fouled-spark-plug` are observed, then the scenario in Figure 2 is an explanation of the observations. The scenario suggests that the culprit of the observed symptoms is `too-rich-fuel-mix` and thereby solves the diagnosis problem.

2.2. Plan recognition

Representing and reasoning about plans is one of the central concerns of AI. Plans are sequences of actions that achieve a goal. The problem of generating a sequence of actions to accomplish a goal is referred to as *plan synthesis.* Plan recognition is the inverse problem of plan synthesis, that of inferring the apparent goal of an agent (or agents) from an observed sequence of actions performed by the agent (or agents).

In cases where we have incomplete specifications of how agents act in environments, the motivation for plan recognition is to somehow attribute a plan to the agent's observed actions, in order to help create a less incomplete specification of behaviour. The extra information, regardless of how it is fabricated, is typically used to do things like explain actions already taken, or to predict actions to be taken. So, for example, just assuming that an acting agent *has* an implicit plan to be recognized provides a basis for a recognizing agent to fabricate and act on the observed agent's intentions, beliefs, and goals. Simply, and intuitively, attributing a plan to an agent lets us ask questions like "why did the agent do that?" and "what will the agent do next?"

Existing applications of plan recognition are in discourse understanding [40], and intelligent interface [6, 22]:

Discourse Understanding: A discourse refers to a sequence of sentences in a episode of dialog or a story. The sentences in a discourse are usually related to each other. To understand a discourse involves recovering the coherent structural relationships among the sentences or components of the sentences. For example, pronoun resolution is a typical task in discourse understanding, which to determine that in the paragraph:

> The man showed the boy a T-shirt. He bought two of them.

It must be inferred that "He" refers to "the boy" and "them" refers to a collection of T-shirts.

The use of a plan to represent the actions and propositions mentioned in the sentences turns the discourse understanding problem into a plan recognition problem [41].

Intelligent Interface: In order to provide more intelligent and more cooperative interfaces to computer systems data bases or expert systems, it is necessary for the intelligent interface to infer the underlying task-related plan that

> motivates the user's queries. For example, consider the following dialogue [53]:
>
> Q: I want to talk to Kathy. Do you know the phone number at the hospital?
> R: She's already been discharged. Her home number is 555-1238
>
> The reason for R's response is that R realizes that Q's query about Kathy's phone number is part of a plan to talk to her. Given what R knows, a direct answer to the question is not going to help Q to achieve his goal. R therefore is able to provide more useful information even though it is not explicitly requested by Q.

Typically plans are hierarchical, and involve subgoals and mediating intentions as well as an ultimate goal. Similar to diagnosis, we use a network called the plan hierarchy to represent domain knowledge about plans. Figure 3 shows a portion of the plan hierarchy for a cooking domain. The nodes of a plan hierarchy represent event types, such as `make-noodle` and `make-pasta`. There are three kinds of links in the network: "isa" links, specialization links, and decomposition links. The "isa" and specialization links represent the taxonomic relationships among the plan types. For example, since `make-spaghetti` is a kind of `make-noodle`, thus, there is an "isa" link from `make-spaghetti` to `make-noodle` and a specialization link from the latter to the former. The decomposition links represent structural relationships among the plan types. For example, since `make-noodle` is a substep of `make-pasta`, there is a decomposition link from `make-pasta` to `make-noodle`. Each decomposition link is labeled with a role function name.

The input to the plan recognizer is a set of observations about the actions of an agent. These actions may be represent by a set of nodes in the plan hierarchy. For example, {`get-spaghetti`, `boil`, `make-marinara`} is a set of observations involving three actions. Given a set of observations, the plan recognizer's task is to infer a higher level goal that explains them. We say a goal explains a set of actions if a plan to accomplish that goal involves executing the set of actions.

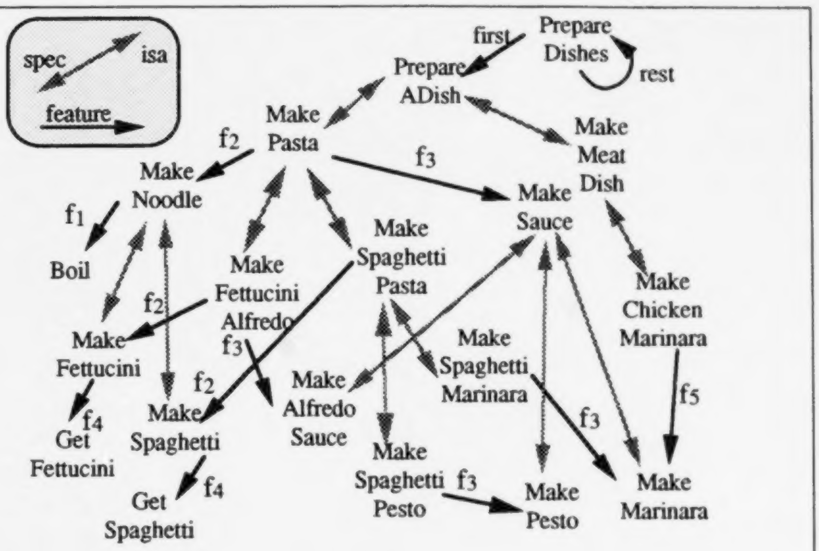

spec
isa
feature
first
Prepare Dishes
rest
Prepare ADish
Make Pasta
Make Meat Dish
f2
f3
Make Noodle
Make Sauce
f1
Boil
Make Fettucini Alfredo
Make Spaghetti Pasta
Make Chicken Marinara
f2
Make Fettucini
f3
Make Spaghetti Marinara
f2
Make Alfredo Sauce
f5
f4
Get Fettucini
Make Spaghetti
f3
Make Spaghetti Pesto
f3
Make Pesto
Make Marinara
Get Spaghetti
f4

Figure 3: The plan hierarchy of a cooking world

In our plan hierarchy, an explanation of a set of observations is represented by a subtree of the plan hierarchy that connects the set of observations to a higher level goal. For example, Figure 4 shows a subtree of the plan hierarchy that explains {`get-spaghetti`, `boil`, `make-marinara`}, because the subtree represents a scenario of `make-spaghetti-marinara` which involves the actions: `get-spaghetti`, `boil`, and `make-marinara`.

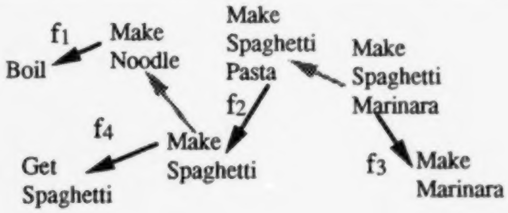

f1
Boil
Make Noodle
Make Spaghetti Pasta
Make Spaghetti Marinara
f2
f4
Get Spaghetti
Make Spaghetti
f3
Make Marinara

Figure 4: A scenario

2.3. Parsing

Many tasks in natural language understanding can be viewed as abductive reasoning. These include both low-level tasks such as word sense disambiguation and high-level tasks such as discourse understanding and compound nominal interpretation [28].

We show that parsing can also be viewed as abduction, where a parse tree of a sentence is taken to be an explanation of how the words in a sentence relate.

2.3.1. Context-free grammar

A context-free grammar consists of six components:

1. The words in the language are denoted by a set of **terminal** symbols.
2. The grammatical categories are denoted by a set of **non-terminal** symbols.
3. The **pre-terminal** symbols are a subset of non-terminals denoting the parts of speech of the words.
4. The **start symbol** is a distinguished non-terminal denoting the sentences in the language.
5. The **phrase-structure rules** (PS-rules) are used to define what word sequences are sentences of the language. The rules are in the form:

   $$R : m \longrightarrow d_0 \ldots d_k$$

   where R is the name of the rule and $m, d_0, \ldots, d_k$ are non-terminals.
6. the **lexicon** of the language is a set of association pairs (w, p), where w is a terminal and p is a set of pre-terminals that are w's parts of speech.

This definition of context-free grammar is equivalent to its usual formulation such as in [29]. Figure 5 shows a simple context-free grammar.

non-terminals = {S, NP, VP, PP, $*n$, $*v$, $*p$, $*d$}
pre-terminals = {$*n$, $*v$, $*p$, $*d$}
terminals = {I, saw, a, man, in, the, park}
start symbol = S
PS-rules =
R_s: $S \longrightarrow NP\ VP$; R_{np1}: $NP \longrightarrow *n$;
R_{np2}: $NP \longrightarrow *d\ *n$; R_{np3}: $NP \longrightarrow NP\ PP$;
R_{vp1}: $VP \longrightarrow VP\ PP$; R_{vp2}: $VP \longrightarrow *v\ NP$;
R_{pp}: $PP \longrightarrow *p\ NP$;
lexicon =
{ (I, {$*n$}), (saw, {$*v$, $*n$}), (a, {$*d$}), (man, {$*n$}),
(in, {$*p$}), (the, {$*d$}), (park, {$*n$}) }

Figure 5: An example CFG grammar

2.3.2. A network representation of CFG

A context-free grammar can be represented by a network, where the each node represents either a non-terminal or a PS-rule. The links in the network are between the nodes representing non-terminals and the nodes representing PS-rules:

> For each rule
> $$R : m \longrightarrow d_0 \ldots d_k$$
> in the grammar, the grammar network has a link from m to R and a link from R to d_i ($i = 0, \ldots, k$). The link from R to d_i is labeled i (Figure 6).

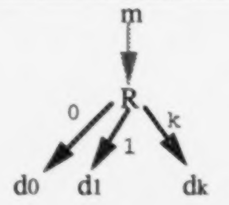


Figure 6: Translation of CFG rules into a network

The network representing the grammar in Figure 5 is shown in Figure 7. There are two types of nodes in the network. The nodes representing non-terminals are called `NT` nodes and those representing phrase structure rules are known as `PSR` nodes. There are also two types of links in the network. A link from an `NT` node to a `PSR` node is called a `OrLink` and a link from a `PSR` node to an `NT` node is called an `AndLink`.

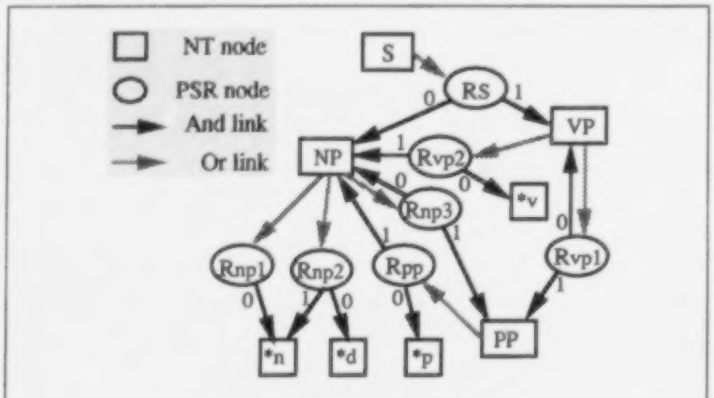


Figure 7: A grammar network

We use integers to represent the word boundaries in the input sentence. Segments of the sentence can then be represented by intervals. For example, suppose the input sentence is `I saw a man in the park`, the interval [1,4] then represents the segment `saw a man`.

`I saw a man in the park.`

$*n$ $*v, *n$ $*d$ $*n$ $*p$ $*d$ $*n$

0 1 2 3 4 5 6 7

Figure 8: The input sentence

A local tree of the grammar network is a `PSR` node of the grammar network together with all the `AndLinks` in the network that are adjacent to it. The nodes of a local tree are annotated with intervals such that the intervals associated with the leaf nodes are disjoint (except at the ends) and the union of the intervals is equal to the interval associated with the root node of the local tree. Figure 9 shows an example of a local tree.

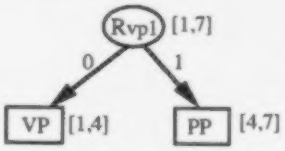


Figure 9: A local tree

A generalized subtree of the grammar network is defined recursively as follows:

1. The node representing the start symbol, annotated with an interval $[0, n]$ is a generalized subtree.

2. Let α be a generalized subtree and p be a leaf node in α annotated with interval $[i, j]$. Let β be an `OrLink` from p to q, or a local tree whose root node is p. If the node p in β is also annotated with the interval $[i, j]$, then the tree formed by joining α and β at node p is also a generalized subtree.

Given an input sentence, the words in the sentence are mapped to a set of annotated nodes in the grammar network by the lexicon. The parsing problem becomes that of finding a generalized subtree of the grammar network that connects the these nodes. The generalized subtree can serve as a parse tree of the input sentence. Figure 10 shows a parse tree of the sentence in Figure 8.

2.4. Summary

The common theme in the three applications of obvious abduction can be summarized as follows: The domain knowledge is represented by a network, whose nodes represent the objects and the links in the network represent the structural and taxonomic relationships between the objects. Observations are represented by nodes in the network. An explanation of a set of observations is a scenario, which is a subtree of the network that connects the observations.

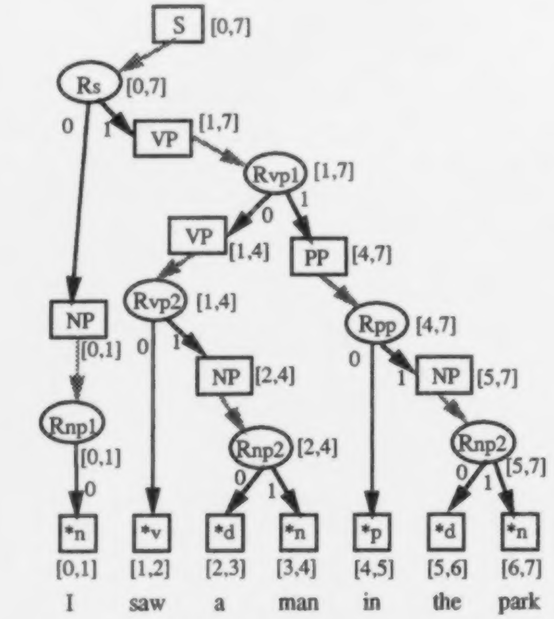

S [0,7]
Rs [0,7]
0
1
VP [1,7]
Rvp1 [1,7]
0
1
VP [1,4]
PP [4,7]
NP
Rvp2 [1,4]
Rpp [4,7]
[0,1]
0
1
NP [2,4]
0
1
NP [5,7]
Rnp1
[0,1]
0
Rnp2 [2,4]
0
1
Rnp2
[5,7]
0
1
*n
*v
*d
*n
*p
*d
*n
[0,1]
[1,2]
[2,3]
[3,4]
[4,5]
[5,6]
[6,7]
I
saw
a
man
in
the
park

Figure 10: A generalized subtree

Chapter 3

Obvious abduction: a formal theory

We will first review fundamental concepts in logic and probability and briefly describe the language **Lp** in the next two sections. We will then present a formal theory of obvious abduction, which contains the following components:

- A representation language.
- A formulation of the abduction problem and its solution.

The knowledge representation language is defined in Sections 3.3 and 3.4. The second issue is dealt with in Sections 3.5 through 3.7.

3.1. Logic and probability

In obvious abduction, both logical and probabilistic knowledge about the domain are represented and used in arriving at the best explanation of the observations. This section briefly reviews the fundamental concepts in semantics of logic and formal probability theory.

3.1.1. Logic

> "The formalization of knowledge in declarative form begins with a *conceptualization.* This includes the objects presumed or hypothesized to exist in the world and their interrelationships [18]"

The set of objects in a conceptualization is called a **universe of discourse**. A **first order conceptualization** is a triple consisting $(\mathcal{U}, F_{\mathcal{U}}, R_{\mathcal{U}})$, where $\mathcal{U}$ is the universe of discourse, $F_{\mathcal{U}} \subset \{f | f : \mathcal{U}^n \mapsto \mathcal{U}, n = 1,2,\ldots\}$ is a set of functions, and $R_{\mathcal{U}} \subset \{r | r \subset \mathcal{U}^n, n = 1,2,\ldots\}$ is a set of relations. We will use the term "conceptualization" to refer to first order conceptualization, since we are not concerned with higher order ones.

Given a first order language $\mathcal{L}$ and a conceptualization $\mathcal{C} = (\mathcal{U}, F_{\mathcal{U}}, R_{\mathcal{U}})$, a **possible world** is a pair $(\mathcal{C}, \mathcal{I})$. The first component is a conceptualization. The second component $\mathcal{I}$ is a function, called an **interpretation**, that maps constant symbols in $\mathcal{L}$ into elements in $\mathcal{U}$, function symbols in $\mathcal{L}$ into functions in $F_{\mathcal{U}}$, and relation symbols $\mathcal{L}$ into relations in $R_{\mathcal{U}}$. An interpretation assigns a truth value to every sentence in $\mathcal{L}$. In other words, every sentence in $\mathcal{L}$ is either true or false in a possible world.

3.1.2. Probability

Although the meaning of probability is a contentious issue, the mathematical model of probability is well defined. This model is based on the notions of **sample space** and **event.** The sample space, denoted by Ω, is the set of all outcomes of an random experiment. For a given sample space Ω, (probabilistic) events are subsets of Ω which form a σ-field. That is, they are closed under complement and countable union.

Given a sample space Ω, and a σ-field E of subsets of Ω (the events), a probability P is a real valued function on E satisfying

1. $\forall e \in E.\ 0 \leq P(e) \leq 1$

2. $P(\Omega) = 1$
3. Let $\{e_1, e_2, \ldots, e_n \ldots\}$ be a finite or a denumerably infinite set of mutually exclusive events, then:

$$P(e_1 \cup e_2 \cup \ldots \cup e_n \ldots) = P(e_1) + P(e_2) + \ldots + P(e_n) \ldots$$

The triplet (Ω, E, P) is called a **probability space.**

3.2. The Language Lp

The knowledge representation language for obvious abduction is a restricted subset of **Lp** a language developed by Bacchus [2] to represent and reason with both logical and probabilistic knowledge. This section presents a rather simplified version of **Lp**. The reader is referred to [2] for details.

Prior to **Lp**, most approaches to combining probability and logic (e.g., [45, 4]) attach a probability number $P(\alpha)$ to a first order logic sentence α, where the probability distribution is over a collection of possible worlds. The meaning of $P(\alpha)$ is the measure of the set of possible worlds in which α is true.

More precisely, Probability Logic [45, 25] defines a probability space (Ω, E, P) where the sample space Ω is the set of possible worlds. Each sentence α determines an event in E, which consists of the set of possible worlds where α is true. $P(\alpha)$ is defined to be the probability of α being true in a randomly selected possible world.

Bacchus [2] argues that such formalisms are not capable of making statistical claims about the universe of discourse, such as "more than 50% of the dogs bark." For example, in Probabilistic Logic [45, 25],

$$P(\forall x.dog(x) \rightarrow bark(x)) > 0.5$$

means that "in more than 50% of the possible worlds, all dogs bark" (Figure 11.1), instead of "more than 50% of the dogs bark in all the possible worlds" (Figure 11.2).

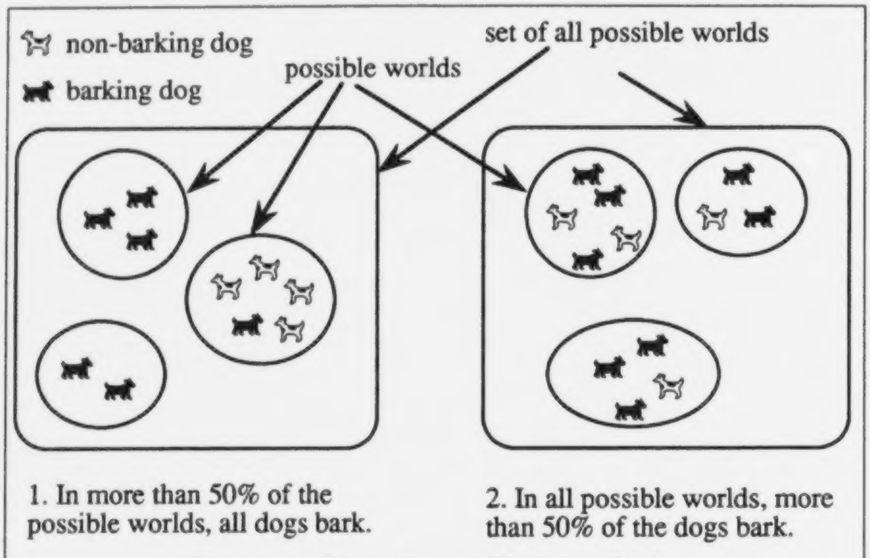

non-barking dog
barking dog
possible worlds
set of all possible worlds
1. In more than 50% of the possible worlds, all dogs bark.
2. In all possible worlds, more than 50% of the dogs bark.

Figure 11: More than 50% of dogs bark

To deal with this problem, Bacchus proposed a language **Lp**[2]. Rather than defining a probability space that consists of all the possible worlds, **Lp** augments each possible world with a probability space. Suppose $(\mathcal{C}, \mathcal{I})$ is a possible world of FOL, where $\mathcal{C} = (\mathcal{U}, F_{\mathcal{U}}, R_{\mathcal{U}})$, **Lp** augments it with a probability space $\mathcal{P} = (\mathcal{U}, E_{\mathcal{U}}, P)$, where the sample space $\Omega = \mathcal{U}$ is the universe of discourse. A possible world of **Lp** is $(\mathcal{C}', \mathcal{I})$, where $\mathcal{C}' = (\mathcal{C}, \mathcal{P})$. This is illustrated in Figure 12. The events in $E_{\mathcal{U}}$ are subsets of the universe of discourse $\mathcal{U}$ and can be defined by **Lp** formulas with one free variable. Suppose $\alpha(x)$ is an **Lp** formula with free variable x, the event defined by $\alpha(x)$ consists of the individuals in $\mathcal{U}$ such that when x is substituted with these individuals, $\mathcal{I}$ interprets the formula as being true. The probability of the event so determined by $\alpha(x)$ is denoted by $[\alpha(x)]_x$.

Intuitively, $[\alpha(x)]_x$ is the probability that $\alpha(x)$ evaluates to *true* if the free occurrences of x in α are substituted by a random individual in the domain $\mathcal{U}$. For example, $[bird(x)]_x$ is the probability of a random individual in the domain

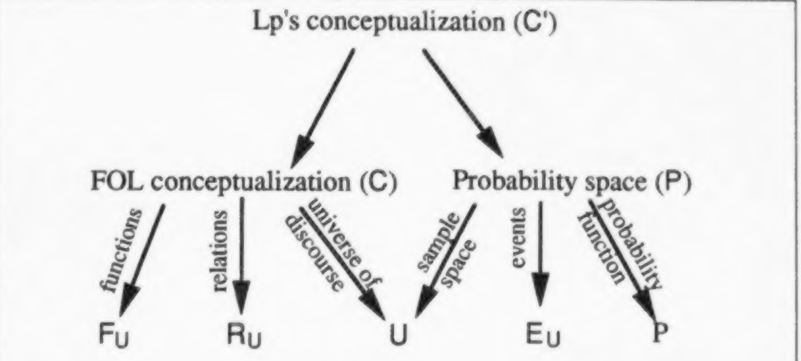

Lp's conceptualization (C')
FOL conceptualization (C)
Probability space (P)
functions
relations
universe of discourse
sample space
events
probability function
FU
RU
U
EU
P

Figure 12: **Lp** augments FOL conceptualization with a probability space

being a bird. Conditional probabilities can also be readily expressed in **Lp** by $[\beta(x)|\alpha(x)]_x$. The semantics of $[fly(x)|bird(x)]_x$ is the probability of a random bird being able to fly.

3.3. Object-Feature-Attribute Representation

3.3.1. Conceptualization

We use a restricted subset of **Lp** as our knowledge representation language for obvious abduction. The restrictions are based on a restricted conceptualization of the world. As mentioned earlier, the conceptualization of a first order language is a triplet $(\mathcal{U}, F_{\mathcal{U}}, R_{\mathcal{U}})$. In our conceptualization, the universe of discourse $\mathcal{U}$ is the union of objects (denoted by $\mathcal{O}$) and their attribute values (denoted by $\mathcal{A}$) such that:

$$\mathcal{U} = \mathcal{O} \cup \mathcal{A}; \quad \mathcal{O} \cap \mathcal{A} = \{\perp\}$$

where $\perp$ is a distinguished element in the universe of discourse that represents the intuitive notion of undefined object or attribute value.

The objects are the individuals in the domain that are of primary interest. For example, in diagnosis, objects are instances of faults/disorders, symptoms or intermediate states; in plan recognition, objects are instances of plans and actions; in parsing, objects are instances of grammatical categories. The attribute values are possible values of the properties, qualities, or characteristics that can be ascribed to the objects.

The set of functions $F_{\mathcal{U}}$ and relations $R_{\mathcal{U}}$ are partitioned accordingly.

Functions on the universe of discourse are partitioned into features F, attributes A, and attribute value functions R. Features are functions that map objects into another objects. Attributes are functions that map objects into attribute values. Attribute value functions are functions over the set of attribute values. Therefore $F_{\mathcal{U}} = F \cup A \cup R$, where $F = \{f | f : \mathcal{O} \mapsto \mathcal{O}\}$, $A = \{a | a : \mathcal{O} \mapsto \mathcal{A}\}$ and $R = \{h | h : \mathcal{A}^n \mapsto \mathcal{A}, n = 1, 2, \ldots\}$.

The feature functions (F) represent structural relationships between objects. For example, a plan to rename a file consists of two steps: first, copy the file, then delete the old copy. The relationships between a plan to rename a file and its two steps are represented by two features: `step1, step2`, which map an instance of `rename-file` to an instance of `copy-file` and `delete-file` respectively (see Figure 13).

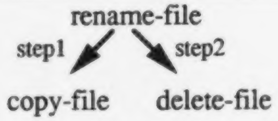


Figure 13: `rename-file` has two steps

The attributes represent the properties or characteristics of the objects. For example, a disorder may have attributes: `patient-name, patient-age, location;` a plan may have attributes: `agent, time;` a grammatical category may have attributes: `case, person, number.`

Attribute value functions are arbitrary arity functions in $\mathcal{A}$. For example, if time intervals are part of A, then `union` is an attribute value function that maps two adjacent or overlapping intervals into another time interval.

The relations $R_{\mathcal{U}}$ in the conceptualization are also partitioned into categories and attribute constraints. A **category** is a unary predicate on the set of objects. The **attribute constraints** are relations on the attribute values $\mathcal{A}$ with arbitrary arity.

Figure 14 illustrates our conceptualization of the world.

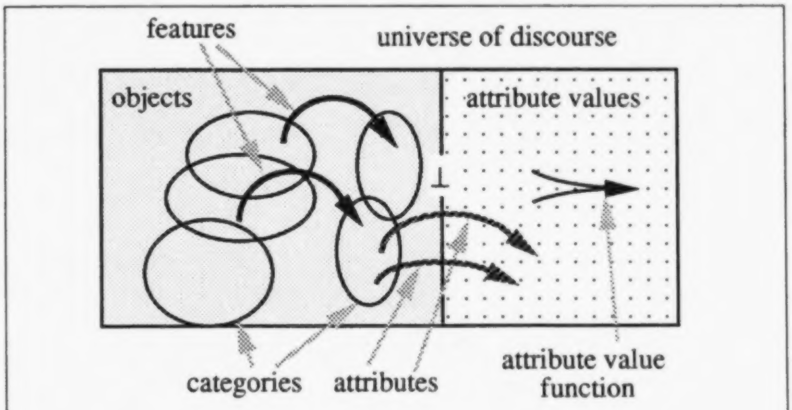


Figure 14: Conceptualization of the world

We also make the following assumptions about the domain:

Assumption 3.1 (Interpretation of $\perp$) Any function with argument $\perp$ evaluates to $\perp$. No predicate holds on $\perp$. Any quantified variable x is implicitly understood to have the qualification $x \neq \perp$.

Assumption 3.2 (Atomic acyclicity) For any object $x \neq \perp$ in the domain, there does not exist a sequence of features $f_1, \ldots, f_n$, such that

$$x = f_n(\ldots f_1(x) \ldots)$$

As we will see in section 5.1 that the acyclicity of values of feature functions reflect the acyclicity of token causation.

3.3.2. Axiomatization

The objects, features and attributes are axiomatized by four sets of **Lp** formulas: Abstraction Axioms, Feature Restriction Axioms, Attribute Value Constraints, and Category Statistics.

Abstraction Axioms define the taxonomic hierarchy among the categories:

Definition 3.3 (Abstraction Axioms) An abstraction axiom is a sentence of the form: $\forall x.c_1(x) \rightarrow c_2(x)$, where c_1, c_2 are categories.

The following are two examples of abstraction axioms:

$$\forall x.\texttt{make-spaghetti-pesto}(x) \rightarrow \texttt{make-pasta}(x)$$

$$\forall x.\texttt{make-spaghetti}(x) \rightarrow \texttt{make-noodle}(x)$$

The "isa" relationships represented by abstraction axioms allow categories to inherit the features of their superclasses. For example, since `make-spaghetti` is subsumed by `make-noodle`, and f_1 is a feature of `make-noodle` representing its boiling step, this feature is inherited by `make-spaghetti`, i.e., an instance of `make-spaghetti` also has a boiling step. Such inheritance is common in semantic networks.

In abduction, there is another kind of inheritance that needs to be considered: the subclass can also inherit explanations of its superclasses. For example, given

> `bad-carburetor-chip` explains `improper-fuel-mix`, and
> `too-rich-fuel-mix` is a kind of `improper-fuel-mix`,

then `bad-carburetor-chip` is also an explanation of `too-rich-fuel-mix`. The explanation is inherited by `too-rich-fuel-mix` from its superclass `improper--fuel-mix`.

To facilitate the inheritance of explanations, the inverse of the "isa" relation is represented explicitly by a feature function, called **specialization**. Corresponding to each abstraction axiom $\forall x.c'(x) \rightarrow c(x)$, there is a specialization function *spec_c* such that

$$spec\_c(x) = \begin{cases} x & \text{if } c(x) \text{ is true} \\ \perp & \text{otherwise} \end{cases}$$

Features are functions over the set of objects. Feature restriction axioms constrain the range of the feature functions.

Definition 3.4 (Feature Restriction Axioms) A feature restriction axiom is a sentence in the form:

$$\forall x. c(x) \wedge (f(x) \neq \perp) \rightarrow c'(f(x))$$

where c, c' are categories and f is a feature function.

We write $c \xrightarrow{f} c'$ as a short hand for this axiom. For example, suppose f_1 is a feature (function) that maps an instance of `make-noodle` to one of its component, say its second step, an instance of `boil`. Then there is a feature restriction axiom

$$\forall x.\texttt{make-noodle}(x) \wedge (f_1(x) \neq \perp) \rightarrow \texttt{boil}(f_1(x)).$$

A feature $f \in F$ is said to be a *feature of a category* c provided that there exists a feature restriction axiom $\forall x. c(x) \wedge (f(x) \neq \perp) \rightarrow c'(f(x))$. For example, f_1 is a feature of `make-noodle`.

If an object x belongs to a category c and f is a feature of c, or of a superclass of c, then $f(x)$ is an *immediate feature* of x. The *features of an object* x are the objects that are either immediate features of x or, recursively, features of the immediate features of x. For example, $\texttt{make-noodle}(f_2(x))$ is an immediate feature of $\texttt{make-pasta}(x)$, and $\texttt{boil}(f_1(f_2(x)))$ is a feature of $\texttt{make-pasta}(x)$.

In abduction, explanations must be consistent with the knowledge base. However, consistency checking is a major source of complexity. In first order logic, consistency checking is semi-decidable. In our approach, the only allowable constraints are the constraints between attribute values of the same object or attribute values of an object and its immediate features. As a result, consistency requirements can be established efficiently by local computation. The following sets of axioms specifies the patterns of constraints in obvious abduction.

Definition 3.5 (Attribute Value Constraints) There are two types of attribute value constraints: local constraints and percolation constraints.

A Local Constraint specifies how the attributes of an object are related, and is a formula in the form of either:

$$\forall x. c(x) \rightarrow \gamma_0(x)$$

or

$$\forall x.(c(x) \wedge \gamma_1(x) \wedge \ldots \wedge \gamma_k(x)) \rightarrow (a(x) = \sigma(x))$$

where c is a category,
$\gamma_j(x) \equiv r(a_{i_1}(x), a_{i_2}(x) \ldots, a_{i_n}(x))$ is a n-ary predicate over attribute values $(\mathcal{A})$ $(n = 0, 1, 2, \ldots)$,
$\sigma(x) \equiv h(a'_{i_1}(x), a'_{i_2}(x) \ldots, a'_{i_k}(x))$ is an attribute value function
a_j and a'_j are attributes.

A Percolation Constraint specifies how an attribute of an object is determined by an attribute of its immediate feature:

$$\forall x, v \,.\, (c_1(x) \wedge c_2(f(x)) \wedge a'(f(x)) = v) \;\rightarrow\; a(x) = h(v)$$

where c_1, c_2 are categories, f is feature of c_1, a, a' are attributes, and h is an attribute value function.

Let f_4 be the feature of `make-spaghetti` that maps an instance of `make-spaghetti` to its `get-spaghetti` component, and f_1 is a feature of `make-noodle`. Let `time`, `time_`f_1, and `time_`f_4 be attributes and `before` be a binary predicate of attribute values. Then

$$\forall x.\texttt{make-spaghetti}(x) \rightarrow \texttt{before}(\texttt{time\_}f_4(x), \texttt{time\_}f_1(x))$$

is an example of local constraints.

Two examples of percolation constraints are:

$$\begin{aligned} \forall x, v.\;\; & \texttt{make-spaghetti}(x) \wedge \texttt{get-spaghetti}(f_4(x)) \wedge \texttt{time}(f_4(x)) = v \\ & \longrightarrow \texttt{time\_}f_4(x) = v \\ \forall x, v.\;\; & \texttt{make-noodle}(x) \wedge \texttt{boil}(f_1(x)) \wedge \texttt{time}(f_1(x)) = v \\ & \longrightarrow \texttt{time\_}f_1(x) = v \end{aligned}$$

Together, the above local constraint and percolation constraints require that the `get-spaghetti` step of `make-spaghetti` happen before the `boil` step.

So far, the axioms have all been first-order logic (FOL) formulas. **Lp**'s extension to FOL is the ability to represent and reason with probabilistic knowledge.

An **attribute assignment** of an object x is a set of assignments of attribute values to x and is denoted by $V(x)$. That is

$$V(x) \equiv \bigwedge_{i=1}^{k} a_i(x) = v_i$$

where a_i is an attribute and v_i is an attribute value ($i = 1, \ldots, k$).

An **object description** is an assertion of the category of an object and an assignment of its attributes. An object description can be written in the form:

$$D(x) \equiv c(x) \wedge V(x)$$

where c is a category and $V(x)$ is an attribute assignment.

Definition 3.6 (Category Statistics) Category Statistics represent the statistical knowledge about the categories. Let f be a feature of c, which maps the objects in c into objects in c'. Let $D(x) = c(x) \wedge V(x)$ and $D'(f(x)) = c'(f(x)) \wedge V'(f(x))$ be object descriptions. Category Statistics are **Lp** formulas in either one of the forms:

$$[D'(f(x)) \mid D(x)]_x = p \quad \text{or} \quad [D(x)]_x = p$$

where $p \in [0, 1]$ is a real number.

For example, [`add-seasoning`($f_6(x)$)|`make-pasta`(x)]$_x$ is the probability that a random instance of `make-pasta` has an `add-seasoning` component. The semantics of the probability term [`make-pasta`(x)]$_x$ is the probability of a random object in the domain being an instance of `make-pasta`.

In summary, the domain knowledge is represented by four sets of **Lp** sentences. We write KB (knowledge base) to denote the conjunction of the formulas in sets of Abstraction Axioms, Feature Restriction Axioms, Attribute Value Constraints and Category Statistics.

3.4. The language $\mathbf{S}^3$

We use a language $\mathbf{S}^3$ ($\mathbf{S}^3$ is Syntactic Sugar) to express axioms in a more concise form. The BNF syntax of $\mathbf{S}^3$ is specified in Figure 15. The meanings of the meta-symbols are as follows:

$\Rightarrow$ separates the right hand side and the left hand side of a rule.

| separates two alternatives of a rule.

[ ] embrace an optional component of a rule.

. ends a rule.

The nonterminal symbols are in small capital font like THIS. We have adopted the naming convention that <SYMBOL>-LIST is a non-empty list of <SYMBOL>'s. For example CATEGORY-LIST is a non-empty list of categories.

An example category definition in $\mathbf{S}^3$ is as follows:

```
(defcategory make-spaghetti-marinara
  (attributes
    :export (agent time)
    :local (f2-time f3-time))
  (isa make-pasta)
  (defeature f2 make-spaghetti
    (percolate agent agent)
    (percolate f2-time time))
  (defeature f3 make-marinara
    (percolate agent agent)
    (percolate f3-time time))
  (local-constraint
    (eq time (union f2-time f3-time))))
```

A category definitions in $\mathbf{S}^3$ is a short hand for a set of **Lp** sentences. The translation is briefly described as follows:

1. ATTRIBUTE-DEF specifies which attributes can be ascribed to the objects in this category. The export attributes are percolated to other categories of which this category is a feature. Other attributes are called local attributes. They are only involved in local constraints.

2. PRIOR-PROB is a function that returns a probability given an object description.

3. Each element in the list **(isa** CATEGORY-LIST**)** is translated into an abstraction axiom. Therefore `(isa make-pasta)` is translated to:

$$\forall x.\texttt{make-spaghetti-marinara}(x) \rightarrow \texttt{make-pasta}(x)$$

```
CATEGORY-DEF  ⇒
    (defcategory CATEGORY
        [ ATTRIBUTE-DEF ]
        [ (isa CATEGORY-LIST )]
        [ FEATURE-DEF-LIST ]
        [ (local-constraint
              LOCAL-CONSTRAINTS-LIST ) ]
        [ PRIOR-PROB ] ) .

ATTRIBUTE-DEF  ⇒
    (attributes
        [ :export ( ATTRIBUTE-LIST ) ]
        [ :local ( ATTRIBUTE-LIST ) ] ) .

FEATURE-DEF  ⇒
    (defeature FEATURE  CATEGORY
        [  CONDITIONAL-PROB ]
        [ PERCOLATION-CONSTRAINT-LIST ] ) .

PERCOLATION-CONSTRAINT  ⇒
    (percolate ATTRIBUTE  ATTRIBUTE-TERM ) .

LOCAL-CONSTRAINT  ⇒
    ATTRIBUTE-PRED  |
    (→ ( ATTRIBUTE-PRED-LIST )
         ATTRIBUTE-EQ ) .

ATTRIBUTE-PRED  ⇒
    ( PRED  ATTRIBUTE-TERM-LIST ) .

ATTRIBUTE-TERM  ⇒
    ATTRIBUTE  |
    ( FUNCTION  ATTRIBUTE-TERM-LIST ) .

ATTRIBUTE-EQ  ⇒
    (eq ATTRIBUTE  ATTRIBUTE-TERM ) .
```

Figure 15: The BNF syntax of $\mathbf{S}^3$

4. Each FEATURE-DEF is translated into a feature restriction axiom and a set of percolation constraints. In the above example, `(defeature f2 make-spaghetti)` is translated into:

   $$\forall x.\texttt{make-spaghetti-marinara}(x) \wedge f_2(x) \neq \perp \rightarrow \texttt{make-spaghetti}(f_2(x))$$

5. CONDITIONAL-PROB is a function that returns a conditional probability given two object descriptions.

6. Each PERCOLATION-CONSTRAINT is translated into a percolation constraint. For example,

   ```
   (defeature f2 make-spaghetti
     (percolate agent agent))
   ```

   is translated into:

   $$\begin{array}{ll} \forall x, v. & \texttt{make-spaghetti-marinara}(x) \wedge \\ & \texttt{make-spaghetti}(f_2(x)) \wedge \\ & \texttt{agent}(f_2(x)) = v \\ & \rightarrow \\ & \texttt{agent}(x) = v. \end{array}$$

7. Each LOCAL-CONSTRAINT is translated into a local constraint. For example,

   ```
   (local-constraint
     (eq time (union f2-time f3-time)))
   ```

   is translated into:

   $$\begin{array}{ll} \forall x. & \texttt{make-spaghetti-marinara}(x) \rightarrow \\ & \quad \texttt{time}(x) = \texttt{union}(\texttt{f2-time}(x), \texttt{f3-time}(x)) \end{array}$$

From now on, we will use $\mathbf{S}^3$ sentences as a short hand for the logical formulas they translate into.

3.5. Scenarios

Abductive reasoning is the inference to the best explanation. We define explanations to be scenarios that entail the observations. A scenario is an **Lp** formula that describes an object and its features. The syntax of scenario is not defined by BNF, rather, it is defined in terms of walk trees in the CF-Diagram which is the diagrammatic form of the abstraction and feature restriction axioms in KB.

Definition 3.7 (CF-Diagram) A Category-Feature diagram (CF-Diagram) is a directed graph whose nodes represent categories and whose links are as follows:

"isa" and specialization links: Corresponding to each abstraction axiom $\forall x.c_1(x) \rightarrow c_2(x)$, there is an "isa" link from c_1 to c_2 with the label "isa", and a specialization link from c_2 to c_1 with the label "spec."

feature links: Corresponding to each feature restriction axiom

$$\forall x.(c_1(x) \wedge f(x) \neq \perp) \rightarrow c_2(f(x))$$

there is a *feature link* from c_1 to c_2 and labeled f. r

We write $\mathrm{CF}_{\mathrm{KB}}$ to denote the CF-Diagram for KB. Figure 16 shows an example of CF-Diagram of the cooking world. Each of the bidirectional gray arrows represents a pair of "isa" and specialization links. The upward arrow represents the "isa" link and the downward arrow represents the specialization link.

The "isa" links induce a partial order ("isa" relation) over the set of categories. The closure of "isa" relation is denoted by $\xrightarrow{\text{isa}}*$ and the non-empty closure is denoted by $\xrightarrow{\text{isa}}+$. The meaning of $\xrightarrow{\text{spec}}*$ and $\xrightarrow{\text{spec}}+$ are closures of the specialization relationships.

The "isa" and specialization links allows a subclass to inherit features and explanations from its superclasses. Similar to inheritance reasoning [78, 2], path preemption ensures that only the most specific information is inherited.

Definition 3.8 (Path Preemption)

Generalization Preemption: A path $c\xrightarrow{\text{isa}}+c_2 \xrightarrow{f} c'$ is preempted if there exists a path $c\xrightarrow{\text{isa}}*c_1 \xrightarrow{f} c_1'\xrightarrow{\text{isa}}*c'$ such that $c_1\xrightarrow{\text{isa}}+c_2$ (Figure 17.a).

Specialization Preemption: A path $c \xrightarrow{f} c_2\xrightarrow{\text{spec}}+c'$ is preempted if there exists a path $c\xrightarrow{\text{spec}}*c_1' \xrightarrow{f} c_1\xrightarrow{\text{spec}}*c'$ such that $c_2\xrightarrow{\text{spec}}+c_1$ (Figure 17.b).

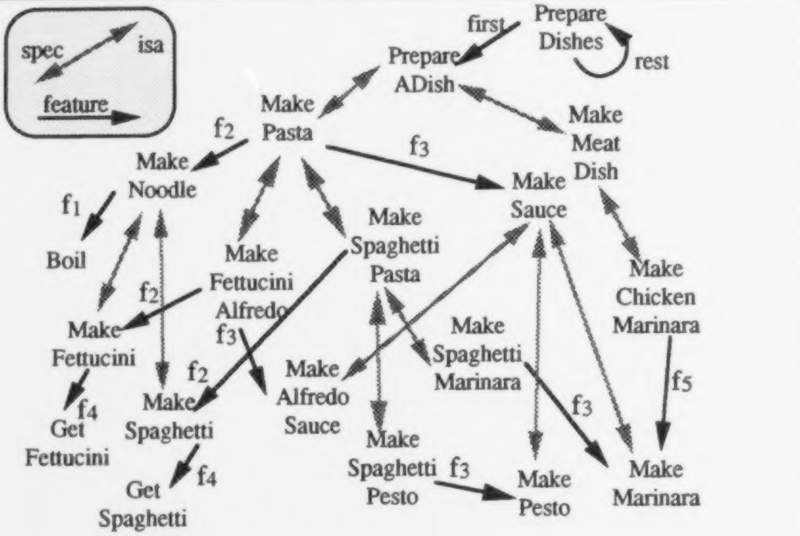
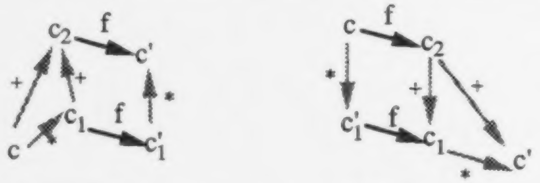

spec
isa
feature
first
Prepare Dishes
rest
Prepare ADish
Make Pasta
Make Meat Dish
f2
f3
Make Noodle
Make Sauce
f1
Boil
Make Fettucini Alfredo
Make Spaghetti Pasta
Make Chicken Marinara
f2
Make Fettucini
f3
Make Spaghetti Marinara
f2
f5
f4
Get Fettucini
Make Spaghetti
Make Alfredo Sauce
f3
Make Spaghetti Pesto
f3
Make Pesto
Make Marinara
Get Spaghetti
f4

Figure 16: The CF-Diagram for the cooking world


c2
f
c'
+
+
*
c
c1
f
c'1
c
f
c2
*
+
+
c'1
f
c1
*
c'

a. Generalization preemption b. Specialization preemption

Figure 17: Path Preemption

An observation can be represented by a node in the CF-Diagram. A directed path in CF_{KB} leading from a node to the observation node hypothesizes that the observation is a feature of the former. Therefore, the path explains the observation. The abductive reasoner's task, however, is not just to explain the observations separately. Rather, it must find a set of coherent relationships which relate all the observations. To accomplish this, we introduce the notion of walk tree. The definition of a walk tree is built recursively upon local trees, which represent the relationships between an object and its immediate features.

Definition 3.9 (Local Tree) A local tree is a one level tree whose the nodes are object descriptions and whose links are labeled with distinct feature functions such that if there is a link from D to D' in the local tree, where $D(x) \equiv c(x) \wedge V(x)$ and $D'(x) \equiv c'(x) \wedge V'(x)$ are object descriptions, then there is a non-preempted path $c \xrightarrow{\text{isa}} * c'' \xrightarrow{f} c'$ in the CF_{KB}.

For example, the tree in Figure 18.1 is a local tree, whereas the tree in Figure 18.2 is not a local tree because the path

$$\texttt{make-spaghetti-marinara} \xrightarrow{\text{isa}} + \texttt{make-noodle} \xrightarrow{f_3} \texttt{make-sauce}$$

is preempted by the path

$$\texttt{make-spaghetti-marinara} \xrightarrow{f_3} \texttt{make-marinara} \xrightarrow{\text{isa}} \texttt{make-sauce}$$

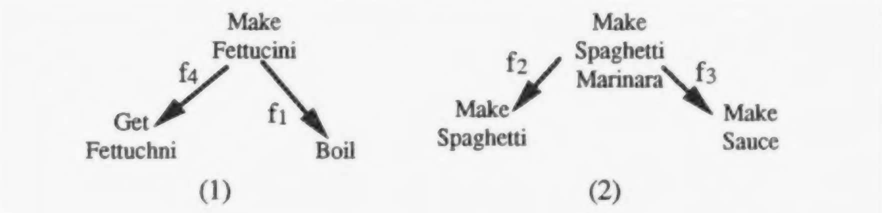


Figure 18: Local tree

Definition 3.10 (Walk Tree) A walk tree is defined recursively as follows:

1. Local trees are walk trees.
2. Suppose α is a walk tree. If D is a leaf node of α and β is a local tree or a specialization link, whose root is also D, then the tree formed by joining the node D in α and the root of β is also a walk tree.
3. All walk trees are either one of the above.

A walk tree is more general than a tree in that there may be multiple occurrences of the same node in CF_{KB}. Therefore, it can be used to represent iterative and recursive plan components in plan recognition or multiple occurrences of the same disorder/symptom in diagnosis. For example, Figure 19 shows a walk tree, where there are two instances of `PrepareADish`.

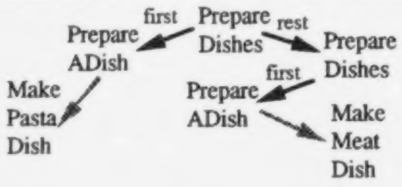


Figure 19: A walk tree

A scenario is a description of a set of related objects. There exists an object in a scenario, called the root, such that all the other objects in the scenario are features of the root.

Definition 3.11 (Scenario) A scenario is a walk tree T. Let $D_1, D_2, \ldots, D_m$ be the nodes in T, where $D_i(x) \equiv c_i(g_i(x)) \wedge V_i(g_i(x))$ is an object description ($i = 1, \ldots, m$) and g_i is the composition of the feature and specialization functions along the path from the root of T to the node D_i ($i = 1, \ldots, m$). Then a scenario $\alpha(x)$ is an **Lp** formula:

$$\alpha(x) \equiv \bigwedge_{i=1}^{m} D_i(g_i(x))$$

The probability of the scenario $\alpha(x)$ is

$$[\alpha(x)]_x = [\bigwedge_{i=1}^{m} D_i(g_i(x))]_x$$

We will show later this probability can be computed from the category statistics.

For example, suppose `time` is an attribute of plans. Let $\alpha(x)$ denotes the scenario in Figure 20, where the intervals beside the nodes denote the values of their `time` attribute. Then $\alpha(x)$ is the following formula:

$$\begin{aligned} \alpha(x) \equiv\ & \texttt{make-spaghetti-marinara}(x) \wedge \texttt{time}(x) = [0,2] \wedge \\ & \texttt{make-marinara}(x) \wedge \texttt{time}(f_3(x)) = [1,2] \wedge \\ & \texttt{make-spaghetti}(f_2(x)) \wedge \texttt{time}(f_2(x)) = [0,2] \wedge \\ & \texttt{get-spaghetti}(f_4(f_2(x))) \wedge \texttt{time}(f_4(f_2(x))) = [0,1] \wedge \\ & \texttt{boil}(f_1(f_2(x))) \wedge \texttt{time}(f_1(f_2(x))) = [1,2] \end{aligned}$$

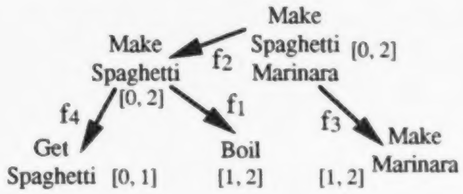


Figure 20: A scenario of `make-spaghetti-marinara`

Since a scenario is tree, a node cannot have two parents. It appears impossible for a scenario to contain interacting causes. Later (p.117), we will show a way to alleviate this problem.

3.6. Abductive reasoning problem

An observation is an assertion that there exists an object that belongs to a certain category and possessing certain attributes, i.e., $\exists x.c(x) \wedge V(x)$.

Definition 3.12 (Observation) An observation is assertion of the existence of an object and is an **Lp** formula in the form:

$$o \equiv \exists x.c(x) \wedge V(x).$$

For example, that the agent got some spaghetti during the time interval $[0, 1]$, boiled it during $[1, 2]$, and made a marinara sauce during $[0, 2]$ can be represented by the following three observations:

$$o_1 \equiv \exists x.\texttt{get-spaghetti}(x) \wedge \texttt{time}(x) = [0, 1]$$
$$o_2 \equiv \exists x.\texttt{boil}(x) \wedge \texttt{time}(x) = [1, 2]$$
$$o_3 \equiv \exists x.\texttt{make-marinara}(x) \wedge \texttt{time}(x) = [0, 2]$$

Definition 3.13 (Explanation) Given KB and a set of observations $o_1, \ldots, o_k$, an explanation of the observations is a scenario $\alpha(x)$ such that

1. $\alpha(x)$ is consistent with KB, i.e., KB $\wedge\, \alpha(x) \not\models false$.
2. KB $\wedge\, \alpha(x) \models o_1 \wedge \ldots \wedge o_k$.

Suppose the observations are o_1, o_2, o_3 in the above example, then the scenario in Figure 20 is an explanation of the observations.

Definition 3.14 (Abductive reasoning problem) Given KB and a set of observations $o_1, \ldots, o_k$, the abduction problem is to find the most probable explanation $\alpha(x)$ of $o_1, \ldots, o_k$

As we have mentioned in Section 1.2, one of the most important question to be answered by a computational account of abduction is the definition of explanation and the criteria for ranking them. In our formulation of the abduction problem, the consistency with the knowledge base and the logical entailment of the observations are necessary conditions of explanations; probability is used to rank the explanations. The definition of explanation has also implicitly incorporated some other factors listed in the subsections 1.2.1 and 1.2.2.

Causality: In a scenario, the relationships between the objects are not simply logical entailment. The features represent the structural relationships. When such relationships are causal, the scenario is a description of a causal evolution process. When features represent the part–whole relationship, the scenario shows how the objects fit together.

Coherence: We treat coherence as a qualitative instead of quantitative property of explanations. An explanation is coherent if it shows the relatedness of the observations. We note that the definitions of quantitative coherence [76, 44] use heuristic measurements.

Specificity: Explanations are maximally specific in the sense that they do not contain preempted paths.

Alternative explanation: The quality of an explanation does not only depend on its own properties, but also on the quality of alternative explanations. We will show in the next chapter that our abductive reasoning algorithm not only finds the most probable explanation, but also the second, third, and so on, most probable explanations. The advantage of being able to calculate the probability of an explanation is that we can ignore the explanations whose probabilities are significantly lower than others. When the alternative explanations are mutually exclusive and exhaustive, the certainty of the alternatives can be measured by the entropy value

$$\sum_{i=1}^{n} p_i \log(p_i^{-1}).$$

Relevance: The hypotheses in an explanation are also the conclusions of the abduction. For an explanation to be relevant to an abductive reasoner, its conclusions must be of interest to the abductive reasoner. If the abductive reasoner can provide a set of categories it is interested in, we can require that the explanations only use hypotheses from this set.

Simplicity: A notable difference between our formulation and many other theories of abduction is role of simplicity. We do not explicitly use simplicity as a criterion for ranking explanations. However, in next section we will show that the most probable explanation is the simplest in the sense that it contains a minimal set of feature links.

The only criterion listed in Chapter 1 that is not considered in our definition of explanation is consilience. The consilience of a theory is measured by the breadth of the data the theory explains. In obvious abduction, we assume that the observations to be explained are explicitly listed. Therefore, consilience criterion is not applicable here.

3.7. Probabilities of explanations

Now that the best explanation has been defined to be the most probable explanation, we show that the probabilities of explanations can be computed from the category statistics in the knowledge base. We first discuss the independence assumption that is necessary for the computation.

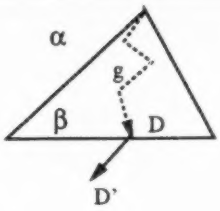


Figure 21: Decomposition of a scenario

Assumption 3.15 (Causation Independence) Let $\alpha(x)$ be a scenario. Let $D \xrightarrow{f} D'$ be a feature link in the scenario such that D' is a leaf node. Let g be the composition of feature and specialization functions along the path from the root node of α to D (See Figure 21). Let $\beta(x)$ be a scenario such that $\alpha(x) \equiv \beta(x) \wedge D'(f(g(x))$. Then we make the following independence assumption:

$$[D'(f(g(x))|\beta(x)]_x = [D'(f(x))|D(x)]_x$$

Any practical application of probability theory involving a large number of events requires independence assumptions of some kind to avoid combinatorial explosion. Many independence assumptions have been criticized for being unrelistic and unreasonable. The independence assumption we made here is based on the structural relationships among the objects in the domain. Intuitively, the assumption states that an object can only influence its features via its immediate features. For example, suppose x is an instance of `make-pasta`, f is a feature that maps x to its `add-seasoning` component. Then, once `make-pasta`(x) is known to be true, the probability of `add-seasoning`($f(x)$) being true is independent of the plans or goals that have lead to `make-pasta`(x). That is, whether `make-pasta`(x) is a component of `make-dinner` or `make-lunch` is irrelevant.

Given the independence assumption, we have the following theorem:

Theorem 3.16 (Probability of Scenario) Let $D_i \xrightarrow{f_i} D'_i$ ($i = 1, \ldots, n$) be the feature links in a scenario $\alpha(x)$. Let D_r be the root, where D_r, D_i, and D'_i

are object descriptions. Then the probability of the scenario $\alpha(x)$ is:

$$[\alpha(x)]_x = [D_r(x)]_x \prod_{i=1}^{n} [D'_i(f_i(x))|D_i(x)]_x$$

Proof: We prove the theorem by induction on the number of feature links in a scenario.

Base Case: The theorem is true of the scenarios that contains no feature links, because if a scenario $\alpha(x)$ only contains one object $D(x)$, then $\alpha(x) = D(x)$. Therefore $[\alpha(x)]_x = [D(x)]_x$. The theorem holds.

Inductive step: Now, suppose the theorem holds for scenarios that contain less than k feature links. Let α be a scenario that contains k feature links. Let $D \xrightarrow{f} D'$ be a feature link in α such that D' is a leaf node in α. Let β be a scenario that is obtained by removing f from α and g be the composition of feature and specialization functions along the path from the root node of α to D. Then,

$$\begin{aligned}
[\alpha(x)]_x &= [\beta(x) \wedge D'(f(g(x)))]_x \\
&= [\beta(x)]_x \times [D'(f(g(x)))|\beta(x)]_x \quad \text{(definition of conditional probability)} \\
&= [\beta(x)]_x \times [D'(f(x))|D(x)]_x \quad \text{(by independence assumption)} \\
&= [D_r(x)]_x \textstyle\prod_{i=1}^{n} [D'_i(f_i(x))|D_i(x)]_x \quad \text{(by inductive assumption)}
\end{aligned}$$

Since the probabilities in the right hand side of the equation are available in the form of Category Statistics, the probabilities of scenarios can thus be computed.

For example, let

$$\begin{aligned}
D(x) &\equiv \texttt{make-spaghetti-marinara}(x) \wedge \texttt{time}(x) = [0,2] \\
D_1(x) &\equiv \texttt{boil}(x) \wedge \texttt{time}(x) \\
D_3(x) &\equiv \texttt{make-marinara}(x) \wedge \texttt{time}(x) = [1,2] \\
D_2(x) &\equiv \texttt{make-spaghetti}(x) \wedge \texttt{time}(x) = [0,2] \\
D_4(x) &\equiv \texttt{get-spaghetti}(x) \wedge \texttt{time}(x) = [0,1]
\end{aligned}$$

Then, the probability of the scenario in Figure 20 is:

$$\begin{aligned}[\alpha(x)]_x &= [D(x)]_x \times [D_2(f_2(x))|D(x)]_x \times [D_3(f_3(x))|D(x)]_x \times \\ & \quad [D_1(f_1(x))|D_2(x)]_x \times [D_4(f_4(x))|D_2(x)]_x\end{aligned}$$

If we associate a weight $-\log([D'_i(f_i(x))|D_i(x)]_x)$ with the link $D_i \xrightarrow{f_i} D'_i$ in a scenario $\alpha(x)$, then $-\log([\alpha(x)]_x)$ becomes the sum weight of its links plus the weight of its root $-\log([D(x)]_x)$. Since maximizing $[\alpha(x)]_x$ is equivalent to minimizing $-\log([\alpha(x)]_x)$, the abduction problem becomes that of finding the scenario with minimal weight.

We say a scenario $\alpha(x)$ is properly contained by another scenario $\beta(x)$ if the If the walk tree of $\alpha(x)$ is a subgraph of the walk tree of $\beta(x)$, and the sum weight of the links in $\beta(x)$ that are not in $\alpha(x)$ is greater than 0. The truth of the following proposition can easily be seen:

Proposition 3.17 The most probable explanation must be the simplest in the sense that it must not be properly contained by another explanation.

3.7.1. A special case

If the set of attributes is empty, the category statistics are of the form $[c(x)]_x$ or $[c'(f(x))|c(x)]_x$, where c, c' are categories and $c \xrightarrow{f} c'$ is a feature or specialization link in the CF-Diagram. We associate each feature or specialization link with a weight $[c'(f(x))|c(x)]_x$ and each node with a weight $[c(x)]_x$. The weight of a subtree of the CF-Diagram is the sum weight of the links in the tree plus the weight of the root. The observations can be represented by a set of nodes in the CF-Diagram. The abduction problem becomes that of finding a subtree of the CF-Diagram that connects all the observation nodes and has the minimum total weight. The later problem is a variation of the Steiner Problem in Graphs, which can be stated as follows:

Definition 3.18 (Steiner Problem in Graphs) Let $G = <N, E>$ be a weighted graph, where N is the set of nodes and E is the set of edges. Each edge $e \in E$

is associated with a non-negative weight $w(e)$. Given a set of nodes $S \subseteq N$, the Steiner Problem in Graphs is to find a sub-tree $T \subseteq E$, such that,

a) all nodes in S are connected together by T;

b) $\sum_{e \in T} w(e)$ is minimal.

The minimal tree is called the Steiner Tree (connecting S).

The importance of this reduction is that the Steiner Problem in Graphs is well-known in graph theory and has been extensively studied. The abduction algorithm presented in the next chapter is inspired and is a generalization of a message passing algorithm for the Steiner Problem in Graphs.

Chapter 4

A generic message passing algorithm

Since only a restricted subset of **Lp** is used to represent the observations, scenarios, and the domain knowledge, the problem of obvious abduction may be solved by a specialized inference algorithm. We present a message passing algorithm for obvious abduction. The next section is an overall description of the algorithm. A detailed account of an object-oriented implementation is given in Section 4.2. The correctness and complexity of the algorithm are discussed in Sections 4.3 and 4.4 respectively.

4.1. Overall description

In the message passing algorithm for obvious abduction, the nodes in the CF-Diagram are also computing agents which communicate by sending messages. Each message contains an item.

Definition 4.1 (Item) An item is a triple: $<O, V, F>$, where O is the subset of the observations to be explained, V is an attribute assignment, and F is a set of features. We call O, V, and F the O-component, V-component, and F-component of the item respectively.

The messages are passed in the reverse direction of the links in CF-Diagram. An item $<O, V, F>$ at a node c means that the observations in O can be explained as features of an object that satisfies the description $c(x) \wedge V(x)$. Therefore, the items represent partial explanations of the observations. The partial explanations are combined at the nodes to generate explanations for larger subsets of the observations.

The observations are represented by items whose F-component is an empty set. Let $\{o_i \equiv \exists x.c_i(x) \wedge V_i(x)\}$ $(i = 1, 2, \ldots, n)$ be a set of observations. The message passing process is initiated by sending the item $<\{o_i\}, V_i, \emptyset>$ to the node c_i, $(i = 1, 2, \ldots, n)$. We call such items original items.

When an item $<O, V, F>$ is sent across a feature link f, the item is translated by the feature link. The result of the translation is another item $<O, V', \{f\}>$, where V' is determined by V according to the percolation constraints. Therefore, the F-components of the items received by a node are either the empty set (in the case of initial messages) or a singleton set. If the item received by a node satisfies the local constraint of the node, it is inserted into the item store of the node. The newly received item is combined with items that are already in the local store. The combination of two items $<O_1, V_1, F_1>$ and $<O_2, V_2, F_2>$ is an item $<O, V, F>$, where $O = O_1 \cup O_2$, V is the unification of V_1 and V_2, and $F = F_1 \cup F_2$. If $<O, V, F>$ satisfies the local constraints, it is also saved into the item store.

The message passing process stops when no further messages are being sent and the explanations can then be retrieved from the network by tracing the origins of the messages whose O-components contain all the observations. The consistency of the explanations is guaranteed by the enforcement of the percolation and local constraints during the message passing process.

The message passing algorithm uses a precomputed relation `preempted` $\subset L \times C$ to ensure that the explanations does not contain preempted paths, where

L is the set of feature links and C is the set of categories. Let l be a feature link $c_1 \xrightarrow{l} c_2$ and c be a category. The binary relation `preempted` is defined as follows: `preempted`(l, c) is true iff 1) the path $c \xrightarrow{\text{isa}}{}^{+} c_1 \xrightarrow{l} c_2$ is preempted; or 2) the path $c_1 \xrightarrow{l} c_2 \xrightarrow{\text{spec}}{}^{+} c$ is preempted.

Once computed, the relation `preempted` remains unchanged as long as the CF-Diagram is unchanged. A message sent across an "isa" link $c' \xrightarrow{\text{isa}} c$ is blocked if `preempted`(l, c) is true, where l is the last feature link traversed by the message. A message sent across a feature link l is blocked if `preempted`(l, c) is true, where c is the last node traversed by the message before reaching l via a sequence of specialization link.

4.2. Object-oriented implementation

In the object-oriented paradigm, we organize software as a collection of discrete objects that incorporate both data members and a set of operations. The objects interact with each other by passing messages. A message is a request for an object to carry out one of its operations. Objects with similar data members and operations are grouped into classes. In some object-oriented programming systems, every object belongs to a class. Therefore, the data members and operations of objects are specified in the class definitions.

The nodes and links in the CF-Diagram are implemented as objects. They belong to the classes `Node` and `Link` respectively.

A `Node` object contains the following data members:

`featureLinks`	the incoming feature links
`isaLinks`	the incoming "isa" links
`specLinks`	the incoming specialization links
`itemStore`	a data structure for storing items

The operations of `Node` include: `sendViaFeature`, `sendViaIsa`, `sendViaSpec`, `receive`, and `receiveViaSpec`. The pseudo C++ code of the operations is shown in Figure 22, where:

- The double colon operator `::` is the scope operator. For example,

  ```
  Node::sendViaFeature(Item i, Node n){ ... }
  ```

  is a definition of the `sendViaFeature` operation of the class `Node`. The operations and data members within this definition are implicitly operations and data members of `Node`.

- The dot operator `.` is the operation and data member accessor of objects. For example, `itemStore.insert(i)` is a request for the object `itemStore` to perform the operation `insert` with parameter `i`.

- Within the definition of an operation, `this` refers to the executer of the operation.

The meanings of the operations are explained as follows:

`sendViaFeature` Send an item across the incoming feature links. The parameter n is the last node before the item i reached this node via a sequence of zero or more specialization links. The preemption relation is checked. The path that consists of feature link l and the sequence of specialization links is preempted if `preempted`(l, n) is true. If the path is not preempted the item $i \equiv <O, V, F>$ is then translated into $<O, V', \{l\}>$, where V' is determined by V according to the percolation constraints. The resulting item is sent to the node at the tail of the feature link.

`sendViaIsa` Send an item across the incoming "isa" links if the path is not preempted.

`sendViaSpec` Send an item across the incoming specialization links.

`receive` Receive an item sent from an outgoing feature link or "isa" link. The item is first inserted into the item store at the node. If the item is new and satisfies the local constraints, it is forwarded across the incoming "isa" links.

```
Node::sendViaFeature(Item i, Node n)
{
  for each l in featureLinks do
    if (not preempted(l, n)) {
      item = l.filter(i);
      l.tail().receive(item, l);
    }
}
Node::sendViaIsa(Item i, FeatureLink f);
{
  for each l in isaLinks do {
    if (not preempted(f, l.tail()))
      l.tail().receive(m, l);
  }
}
Node::sendViaSpec(Item i, Node n)
{
  for each l in specLinks do
    l.tail().receiveViaSpec(m, n);
}
Node::receive(Item i, Link b)
{
  if (not itemStore.valid(i)) return;
  sendViaIsaLinks(i, b);
  newItems = itemStore.combine(i);
  for each item in newItems do {
    sendViaFeature(item, nil);
    sendViaSpec(item, this);
  }
}
Node::receiveViaSpec(Item i, Node b)
{
  if (itemStore.insert(i)) {
    itemStore.insert(i);
    sendViaSpec(i, b);
    sendViaFeature(i, b);
  }
}
```

Figure 22: The operations of `Node` class

The item is then combined with other items in the item store. New items that result from the combinations are sent across feature and specialization links.

`receiveViaSpec` Receive an item sent from an outgoing specialization link. The item is forwarded via the incoming specialization and feature links.

```
ItemList itemStore::combine(Item item)
{
  ItemList newItems, compatibles;
  if (int is_new = insert(item) and complete(item))
    newItems.apend(item);
  compatibles = compatibleItems(item);
  for each i in compatibles do {
    newItem = unify(item, i);
    if (valid(newItem) and insert(newItem) and complete(item))
      newItems.append(newItem);
  }
  return newItems;
}
```

Figure 23: The `add` operation of `ItemStore` class

In the `receive` operation of `Node` class the local processing is delegated to its data member `itemStore` by requesting it to perform the `add` operation. The pseudo code for `add` operation is shown in Figure 23. The `add` operation uses several other generic operations of `ItemStore` class. The generic operations are defined by subclasses of `ItemStore`. To apply the abduction algorithm in different domains amounts to defining different subclasses of `ItemStore` class that implement these generic operations. The semantics of the generic operations are as follows:

`unify(Item i, Item i')` Suppose $i \equiv <O, V, F>$ and $i' \equiv <O', V', F'>$. They are unifiable if O and O' are disjoint, V and V' are unifiable, and F and F' are disjoint. The unification is the item $<O \cup O', V'', F \cup F'>$, where V'' is the unification of V and V'.

`insert(Item i)` If the item `i` is new and satisfies the local constraints, insert the item into the item store and return true. Otherwise, return false.

`compatibleItems(Item i)` Return a list of items in the item store that are compatible with the item i.

`valid(Item i)` Return true if the item i satisfies all the local constraints of the node where this item store is located.

`complete(Item i)` Return true if the F-component of item i contains all necessary features of c, where c is the category the node represents.

The implementations of the operations in different applications domains may take advantage of the specific item structure in that particular domain so that for efficient storage and retrieval. For example, when used in CFG parsing, the observations to be explained are not simply set of words, rather, they are sequences of words. A set of n elements has 2^n subsets. A sequence of n elements, however, has only $O(n^2)$ continuous subsequences. Because of this, as we will see in Section 5.3, the complexity of CFG parsing is polynomial in the number of observations (the number of words in the input sentence).

4.2.1. An example

To illustrate the message passing algorithm, consider the following example: Suppose the domain is the cooking world represented by the CF-Diagram in Figure 16, and the observations are:

$$o_1 \equiv \exists x.\texttt{get-spaghetti}(x) \wedge \texttt{time}(x) = [0,1]$$
$$o_2 \equiv \exists x.\texttt{boil}(x) \wedge \texttt{time}(x) = [1,2]$$
$$o_3 \equiv \exists x.\texttt{make-marinara}(x) \wedge \texttt{time}(x) = [0,2]$$

The execution of the algorithm is as follows:

1. The first observation o_1 is represented by an item

   $$i_1 \equiv <\{o_1\}, \texttt{time} = [0,1], \emptyset>$$

   This item is sent to the node `get-spaghetti`, which saves the item in local store and sends it to `make-spaghetti`.

2. When the node `make-spaghetti` receives item:

   $$i'_1 \equiv <\{o_1\}, \texttt{time} = [0,1], \{f_4\}>$$

   it is inserted into the item store at the node and then sent to the nodes `make-noodle` and `make-spaghetti-pasta` via a specialization and a feature link respectively.

3. When the item i'_1 is received by the node `make-noodle`, it is inserted into the item store at the node and then sent to the node `make-pasta` via the feature link f_2. However, since `preempted`(l,`make-spaghetti`) is true, where l is the feature link:

   $$\texttt{make-pasta} \xrightarrow{f_2} \texttt{make-noodle}$$

   the message to `make-pasta` is blocked.

4. The second observation o_2 is represented by an item

   $$i_2 \equiv <\{o_2\}, \texttt{time} = [1,2], \emptyset>$$

   This item is sent to the node `boil`.

5. The item i_2 is sent to `make-noodle` via the feature link f_1. `make-noodle` receives

   $$i_3 \equiv <\{o_2\}, \texttt{time} = [1,2], \{f_1\}>$$

6. The item i_3 is sent to the node `make-spaghetti` via an "isa" link. The item is combined with i'_1 at `make-spaghetti` and their combination is

   $$i_4 \equiv <\{o_1, o_2\}, \texttt{time} = [0,2], \{f_4, f_1\}>$$

7. The item i_4 is sent to the node `make-spaghetti-pasta` via the feature link f_2. The message is forwarded to the node `make-spaghetti-marinara` by the node `make-spaghetti-pasta`.

8. The third observation o_3 is represented by an item

$$i_5 \equiv <\{o_3\}, \texttt{time} = [0,2], \emptyset>$$

   This item is sent to the node `make-marinara`.

9. The node `make-marinara` sends the message to `make-spaghetti-marinara` via feature link f_3. The item is then combined with i_4 at the node and their combination is

$$i_6 \equiv <\{o_1, o_2, o_3\}, \texttt{time} = [0,2], \{f_2, f_3\}>$$

   which represents an explanation of the three observations.

The message passing process stops, the explanation of the observations $\{o_1, o_2, o_3\}$ can be retrieved by tracing the origins of the item i_6 at node `make-spaghetti-marinara`. That is, the explanation of the observations is a trace of the item whose *O*-component contains all the observations.

4.2.2. Trace of an item

A trace of an item is an annotated tree specifying how the item is generated as the result of a series of combinations and propagations of the original items.

Example 4.2 Let $\{o_1, o_2, o_3\}$ be the three observations in the above example. Then Figure 24 shows a trace of the item

$$<\{o_1, o_2, o_3\}, \texttt{time} = [0,2], \{f_2, f_3\}>$$

at the node `make-spaghetti-marinara`. This item is the unification of the following two items:

$<\{o_1, o_2\}, \texttt{time} = [0,2], \{f_2\}>$ from the node `make-spaghetti`
$<\{o_3\}, \texttt{time} = [1,2], \{f_3\}>$ from the node `make-marinara`.

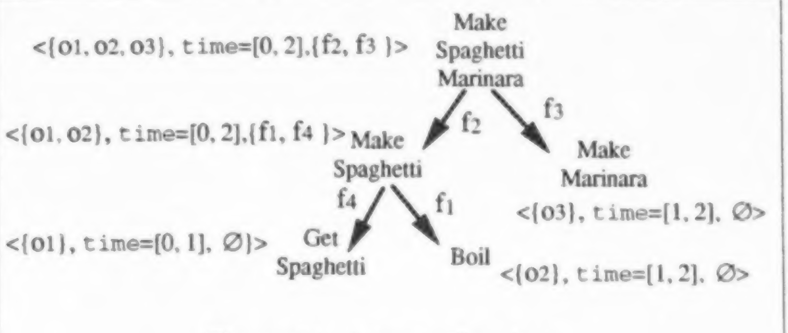

Make Spaghetti Marinara
<{o1, o2, o3}, time=[0, 2],{f2, f3 }>
f2
f3
Make Spaghetti
<{o1, o2}, time=[0, 2],{f1, f4 }>
Make Marinara
<{o3}, time=[1, 2], ∅>
f4
f1
Get Spaghetti
<{o1}, time=[0, 1], ∅}>
Boil
<{o2}, time=[1, 2], ∅>

Figure 24: A trace of an item

The first item can further be traced back to the combination of two items from the nodes `boil` and `get-spaghetti`.

Definition 4.3 (Trace) A trace of an item is an annotated tree, which is defined recursively as follows:

1. For an original item, the trace consists of a single node where the item is located and annotated with the item itself.
2. Suppose an item i located at node c is the unification of k other items i_j $(j = 1, \ldots, k)$ with the singleton F-component. Suppose i_j is a result of sending an item i'_j to node c via a feature link f_j. The trace of i is a tree whose root node is c. The root node has k subtrees, each of which is a trace of i'_j and is connected to the node c by a link labeled f_j.

We interpret a trace as a logical formula with one free variable. Let $c_1, c_2, \ldots, c_n$ be nodes in the trace. Let V_i be the V-component of the item annotating the node c_i and g_i be a compositions of the features along the path from the root of the trace to node c_i. Then the trace is interpreted as a formula:

$$\bigwedge_{i=1}^{n} c_i(g_i(x)) \wedge V_i(g_i(x))$$

4.3. Correctness of the algorithm

The correctness of the algorithm consists of its soundness and completeness. The algorithm is sound if the traces retrieved after the message passing process stops are indeed explanations of the observations. The algorithm is complete if the most probable explanation of the observations is one of the traces of the items. We first establish the soundness of the algorithm.

Theorem 4.4 (Trace consistency) A trace of an item is consistent with the knowledge base.

Proof: A formula is consistent with the knowledge base if we can construct a model of the conjunction of the formula and the knowledge base. We prove the theorem by induction on the number of feature links in the trace.

Base case: Suppose the trace α contains a single node that is interpreted as $D(x) \equiv c(x) \wedge V(x)$. We construct a model where there is only one object o in the domain. The interpretation of the predicates and functions are as follows:

1. For any category c' such that $c \overset{\text{isa}}{\longrightarrow} {*}c'$, $c'(o)$ is true.
2. For any other category c'', $c''(o)$ is false.
3. For any feature function f, $f(o) = \perp$. For any attribute a, if $a(x) = v$ is the consequence of $V(x)$ and the local constraints at c'. Otherwise, $a(o) = \perp$.

Then this model is consistent with the knowledge base iff it is consistent with the local constraints at node c in the CF-Diagram. For all the other local constraints and feature restriction axioms the left hand side of these axioms evaluate to false by the interpretation function. Therefore, they are also true in the model.

Induction step: Let α be a trace, c be the root node of the trace, T be the top level local tree of α, $\alpha_1, \ldots, \alpha_m$ be the subtrees of α, whose roots are leaves of T, $c_1, \ldots, c_m$ (See Figure 25). Then each α_i is a trace of an item.

By the inductive assumption, they are consistent with the knowledge base. Let $M_1, \ldots, M_m$ be the models of $\alpha_i \wedge \text{KB}$. Let f_i $(i = 1, \ldots, m)$ be the feature functions between c and c_i. Let o_i be the object in M_i that corresponds to the root nodes of α_i $(i = 1, \ldots, m)$. We construct a model M of $\alpha \wedge \text{KB}$ as follows:

- The domain of M is the union of the domains of $M_1, \ldots, M_m$ plus a new object o that belongs to the category c.

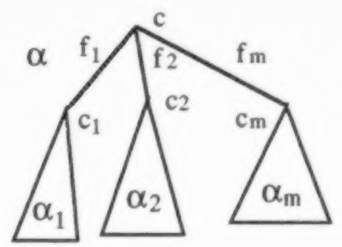


Figure 25: The decomposition of a trace

- M assigns true to $c'(o)$, where $c\xrightarrow{\text{isa}}*c'$. For any other category c'', $c''(o)$ is false. For any f_i $(i = 1, \ldots, m)$, $f_i(o) = o_i$. For any other feature function f, $f(o) =\perp$.
- For any other object o' in the domain, M interprets predicate and function values of o' in the same ways as M_i, where M_i is the domain o' comes from.

Since the model M is constructed according to the formulas in α, the scenario α is obviously true in M. For any axiom $\forall x.\eta(x)$ in the knowledge base, M assigns $\eta(o')$ to be true for any $o' \neq o$ in the model, because the assignment of M is the same as the interpretation function of M_i from which o' comes from. M also assigns true to $\eta(o)$. Therefore, all the axioms in the knowledge base are true in M. Thus, α is consistent with the knowledge base, because M is a model of their conjunction. ∎

Theorem 4.5 (Soundness) A trace of an item is an explanation of the observations in its O-component.

To prove the theorem, we need to show the following:

The trace is a valid walk tree. That is, the trace does not contains a preempted path. If the trace contain a preempted path from $c\xrightarrow{\text{isa}}+c_2 \xrightarrow{f} c'$, then there exists a message that traveled from c' to c_2 to c. However, since $c\xrightarrow{\text{isa}}+c_2 \xrightarrow{f} c'$ is a preempted path, `preempted`(l, c) is true, where l denotes the feature link $c_2 \xrightarrow{f} c'$. The message to c will be blocked by the condition `preempted`$(l, c) = true$ when being passed to c via an "isa" link. By similar argument, we can show that a trace does not contain any preempted path in the form $c \xrightarrow{l} c_2 \xrightarrow{\text{spec}}{}^{+} c'$

The trace logically entails the observations. Since the observations are represented by the original items in the trace, for each observation, there is an object in the trace that matches the description of that observation.

The trace is consistent with the knowledge base. This is the Theorem 4.4 above.

Therefore, the trace of the item is an explanation of the observations in its O-component. ∎

Now that we have proved soundness of the algorithm, we show the completeness of the algorithm. The completeness theorem guarantees that the most probable explanation is a trace of an item.

Scenarios are descriptions of a set of related objects. The completion of a scenario is an augmentation with attribute value assignments that are logical consequences of the original formula.

Definition 4.6 (Scenario completion) Let α be a scenario, then its completion is the closure of the following two rules:

1. If $D_i(g_i(x))$ is an object in the scenario and $a(x) = v$ is a consequence $D_i(x)$ and the local constraints, then $\alpha_{k+1} \equiv \alpha_k \wedge a(g_i(x)) = v$.
2. If $D_i(g_i(x))$ and $D_j(g_j(x))$ are objects in the scenario , $g_i = f \circ g_j$, and $a(x) = v$ is a consequence of $D_j(x) \wedge D_i(f(x))$ and the percolation constraints associated with f, then $\alpha_{k+1} \equiv \alpha_k \wedge a(g_i(x)) = v$.

Since the completion of a scenario involves the same set of objects, the following proposition holds:

Proposition 4.7 The completion of a scenario is also a scenario.

Lemma 4.8 Let $\alpha \equiv \bigwedge_i D_i(g_i(x))$ be a completed scenario, where D_i's are object descriptions and g_i's are compositions of feature functions. Then α entails $\exists x.c(x) \wedge V(x)$ iff there exists D_i such that

1. $D_i(x) \equiv c_i(x) \wedge V_i(x)$,
2. $c_i \overset{\text{isa}}{\rightarrow} *c$, and
3. $V_i(x) \rightarrow V(x)$.

Proof: Suppose there is no such D_i exists in α. We then can use the model construction method described in the proof of Theorem 4.4 to construct a model of $\alpha \wedge \text{KB}$. In this model, none of the objects in the domain satisfies $c(x)$. Thus, $\alpha \wedge \neg\exists x.c(x) \wedge V(x)$ is satisfiable, which implies that α does not explain $\exists x.c(x) \wedge V(x)$. ■

Theorem 4.9 (Completeness) For any explanation α of a set of observations O, there exists an item $<O, V, F>$ at the root node of α, such that the trace β of the item $<O, V, F>$ is an explanation of O, and $\alpha \wedge \text{KB} \models \beta$.

Proof: Let α' be the completion of α. Then, according to the above lemma, for every observation $o_i \equiv \exists x.c_i(x) \wedge V_i(x)$, there is an object description $D_i(x) \equiv c'_i(x) \wedge V'_i(x)$ in α' such that $c'_i \overset{\text{isa}}{\rightarrow} *c_i$ and $V'_i(x) \rightarrow V_i(x)$ $(i = 1, \ldots, k)$. Therefore, during the message passing process, the node c'_i will receive the item

$$<\{o_i\}, V_i, \emptyset>$$

either directly as an original item or from an "isa" link to c_i. Since V_i is weaker formula than V'_i $(i = 1, \ldots, k)$, there will be an item at the root node of α' such that the trace of the item has the same structure of the subtree of α' that spans on D'_i $(i = 1, \ldots, k)$. Let this trace be β, then $\alpha \wedge \text{KB} \models \alpha' \wedge \text{KB} \models \beta$. ■

According to this theorem, the most probable explanation must be a trace of an item that is generated during the message passing process, because the most the probable explanation must not be logically implied by another explanation that has lower probability.

4.4. Complexity of the algorithm

The computational complexity of a *problem* is its intrinsic difficulty as measured by the minimal computational resources, such as time and memory, required for its solution. The complexity of an algorithm is the cost of resources used by the algorithm. The complexity studied here is the algorithm complexity, instead of problem complexity.

Complexity can also be categorized as worst case complexity, average case complexity, probabilistic complexity, etc. Here we are concerned with worst case complexity.

The complexity discussed here is the time complexity in a uniform-cost RAM model. We assume the the following local computations in the message passing algorithm are unit operations:

```
ItemStore::insert
ItemStore::valid
ItemStore::complete
ItemStore::unify
```

The size of the problem are defined by the following parameters:

$|\mathrm{CF}_{\mathrm{KB}}|$: the total number of links in the CF-Diagram;

k: the number of observations to be explained;

d: the maximum number of features a category;

M: the number of possible attribute value assignments of an item.

The message passing process can be divided into two parts: message transmission and local processing. The complexity of the first part is proportional to the number of messages that are transmitted during the message passing process. The complexity of the second part is proportional to the number of pairs of compatible items that are tested for unification. The complexity of the algorithm is the sum of the complexity of the two parts.

Proposition 4.10 No identical messages are sent across a link more than once.

To see that this proposition is true, we observe that executions of the sending methods `sendViaFeature(i, n)`, `sendViaIsa(i, f)`, `sendViaSpec(i, n)` are preceded by the condition that `insert(i)` returns true. When `insert(i)` returns true, the item is new to the node.

The message transmitted across a feature link or an "isa" link is a triple

$$<O, V, \{f\}>$$

where O is a subset of the observations to be explained, V is an attribute value assignment, and f is the last feature link traversed by the message. Therefore, there are at most $O(2^k M)$ distinct messages. The total number of messages that are transmitted during the message passing process is bounded by $O(2^k M|\mathrm{CF_{KB}}|)$.

When a message arrives at a node, it is combined with the items that are compatible with it. For two items

$$i \equiv <O, V, F> \text{ and } i' \equiv <O', V', F'>$$

to be compatible, the intersections $O \cap O'$ and $F \cap F'$ must be empty. Let $\hat{O}$ be the set of all the observations to be explained, and $\hat{F}$ be the set of immediate features of the node where the item i is located. Then O' and F' must be the subsets of $\hat{O} - O$ and $\hat{F} - F$ respectively. Therefore, the item i is compatible with at most $2^{|(\hat{O}\cup\hat{F})-(O\cup F)|}M$ items.

Since no identical items are sent across a link more than once, a node receives at most $2^{|\hat{O}\cup\hat{F}|}M$ items. Each of these items is combined with at most $2^{|(\hat{O}\cup\hat{F})-(O\cup F)|}M$ items. There are $|\mathrm{CF_{KB}}|$ links in the network. Therefore, the total number of unifications attempted is bounded by:

$$
\begin{aligned}
& |\mathrm{CF}_{\mathrm{KB}}| \textstyle\sum_{O \cup F \subset \hat{O} \cup \hat{F}} 2^{|(\hat{O} \cup \hat{F}) - (O \cup F)|} M^2 \\
& = |\mathrm{CF}_{\mathrm{KB}}| \textstyle\sum_{j=1}^{j=|\hat{O} \cup \hat{F}|-1} \sum_{\substack{O \cup F \subset \hat{O} \cup \hat{F} \\ |O \cup F| = j}} 2^{|(\hat{O} \cup \hat{F}) - (O \cup F)|} M^2 \\
& = |\mathrm{CF}_{\mathrm{KB}}| M^2 \textstyle\sum_{j=1}^{j=|\hat{O} \cup \hat{F}|-1} \sum_{\substack{O \cup F \subset \hat{O} \cup \hat{F} \\ |O \cup F| = j}} 2^{|\hat{O} \cup \hat{F}|-j} \\
& = M^2 \textstyle\sum_{j=1}^{j=|\hat{O} \cup \hat{F}|-1} C^j_{|\hat{O} \cup \hat{F}|} 2^{|\hat{O} \cup \hat{F}|-j} \\
& = \left(M^2 \textstyle\sum_{j=0}^{j=|\hat{O} \cup \hat{F}|} C^j_{|\hat{O} \cup \hat{F}|} 2^{|\hat{O} \cup \hat{F}|-j} \right) - M^2 \left(2^{|\hat{O} \cup \hat{F}|} - 1 \right) \\
& = M^2 \left(3^{|\hat{O} \cup \hat{F}|} - 2^{|\hat{O} \cup \hat{F}|} + 1 \right)
\end{aligned}
$$

The last step in the above derivation is based on the equation

$$3^n = (1+2)^n = \sum_{j=0}^{n} C^j_n 2^j$$

Since unification of two compatible items and the insertion of the two items into the item store takes unit time each, the local processing component of the message passing process is bounded by

$$O(M^2 3^{k+d} |\mathrm{CF}_{\mathrm{KB}}|)$$

Therefore the complexity of the message passing algorithm is:

$$O(M^2 3^{k+d} |\mathrm{CF}_{\mathrm{KB}}| + M 2^k |\mathrm{CF}_{\mathrm{KB}}|) = O(M^2 3^{k+d} |\mathrm{CF}_{\mathrm{KB}}|)$$

Chapter 5

Applications of obvious abduction: revisited

In this chapter, we recast the applications of obvious abduction in terms of the theory and the algorithm developed in the previous two chapters. For each application domain, we will show how the domain knowledge is represented in $\mathbf{S}^3$; how problems in these domains can be formulated as obvious abduction.

5.1. Diagnosis

In Section 2.1, we showed informally that diagnosis can be viewed as abduction, where the domain knowledge is represented by a causal network and the diagnosis problem is that of finding a causal scenario that explains the observations. We now show that the causal network can be interpreted as a CF-Diagram.

5.1.1. Causality

Causality is used as a pivotal mechanism in diagnostic reasoning. The explanations of the observed symptoms are expressed in terms of causal relations. The formal definition and characterization of causality is a topic of debate in philosophy[71] and AI [70, 48]. The purpose of this subsection is to clarify our use of the term causation.

Causation is often defined in terms of events. There are two notions of events: **event type** and **token event**. Death is an event type, occurring to different people in different places at different times. Socrates's death is a token event, happening at a unique spatio-temporal location. Event types may alternatively be called general events. Token events are also known as singular events.

Corresponding to the two notions of events, there are two notions of causality:

Causal tendency is a relationship between event types. Causal tendency represents the notion that an event of type X tends to cause another event of type Y and is normally stated when we say "X causes Y."

For example, that "`brain-tumor` causes `coma`" or "`brain-tumor` tends to cause `coma`" is a statement about causal tendency.

Token causation represents the relationship between token events, i.e., an event c actually caused event e. The token causation is normally stated when we say "x (actually) caused y".

For example, "John Smith's brain tumor in his head caused the coma he is in" is a statement about token causation.

Both token events and event types are not the same as probabilistic events. Their relationships may be described as follows: probabilistic events are subsets of the sample space that form a σ-field. A token event is an element in the sample space. An event type is a subset of the sample space. Therefore, an event type can be regarded as a probabilistic event. However, the collection of event types need not be closed under union and complement operation. That is, the union of two event types, though a probabilistic event, is not necessarily an event type.

5.1.2. Logic interpretation of causal network

The causal network is a CF-Diagram. Event types are categories and token events are objects. A causal tendency is represented by a feature function that maps events of one type to events of another type. Token causations are represented by the extensions of the feature functions.

For example, the event type `bad-spark` is a category and is represented by a node in the CF-Diagram. The fact that `bad-spark` causes (or tends to cause) `misfiring` is represented by a feature function `cause-mf` that maps instances of `bad-spark` to the corresponding instances of `misfiring` they caused. This causal relation is stated by a feature restriction axiom:

$$\forall x.\texttt{bad-spark}(x) \wedge \texttt{cause-mf}(x) \neq \perp \rightarrow \texttt{misfiring}(\texttt{cause-mf}(x))$$

The sentence "An instance x of `bad-spark` actually caused an instance of `misfiring`" is represented by the formula

$$\texttt{bad-spark}(x) \wedge \texttt{misfiring}(\texttt{cause-mf}(x)).$$

More generally, let $D(x) \equiv c(x) \wedge V(x)$ and $D'(x) \equiv c'(x) \wedge V'(x)$ be two object descriptions. If there is a feature restriction axiom in the knowledge base

$$\forall x.c(x) \wedge (f(x) \neq \perp) \rightarrow c'(f(x))$$

we call $D(x) \wedge D'(f(x))$ a causation event.

In Section 3.5, we defined a scenario to be the description of a set of related objects. For example:

$$\begin{array}{l} \texttt{too-rich-fuel-mix}(x) \wedge \\ \texttt{stall-when-hot}(f_1(x)) \wedge \\ \texttt{incomplete-combustion}(f_2(x)) \wedge \\ \texttt{black-smoke}(f_2(f_3(x))) \end{array}$$

is a scenario in Figure 2, where `too-rich-fuel-mix` caused `stall-when-hot` and `incomplete-combustion`, which in turn caused `black-smoke`. This scenario can be alternatively written as:

$$\alpha_1(x) \wedge \alpha_2(x) \wedge \alpha_3(f_2(x))$$

where

$$\begin{aligned}
\alpha_1(x) &\equiv \texttt{too-rich-fuel-mix}(x) \wedge \texttt{stall-when-hot}(f_1(x)) \\
\alpha_2(x) &\equiv \texttt{too-rich-fuel-mix}(x) \wedge \texttt{incomplete-combustion}(f_2(x)) \\
\alpha_3(x) &\equiv \texttt{incomplete-combustion}(x) \wedge \texttt{black-smoke}(f_3(x))
\end{aligned}$$

are causation events. Therefore, a scenario can also be regarded as a set of chained causation events which constitute a causal process.

A causation event $D(x) \wedge D'(f(x))$ is also a probabilistic event. Its conditional probability given the cause event $D(x)$ is

$$[D(x) \wedge D'(f(x))|D(x)]_x = [D'(f(x))|D(x)]_x.$$

Recall in Section 3.7, we showed that the probability of a scenario $\alpha(x)$ is

$$[\alpha(x)]_x = [D(x)]_x \prod_{i=1}^{n} [D'_i(f_i(x))|D_i(x)]_x$$

where $D(x)$ is a description of the root object in the scenario $\alpha(x)$, i.e., the culprit of the causal scenario.

$D_i \xrightarrow{f_i} D'_i$ $(i = 1, \ldots, n)$ are feature links in the scenario.

Therefore, the probability of a causal scenario is the product of the prior probability of the culprit and the conditional probabilities of the causation events involved in the scenario given their causal events.

5.1.3. Causal independence

In a causal network, the independence assumption we made in Section 3.7 corresponds to our intuitive notion of causality.

Let $\alpha(x)$ is a scenario. Let $D \xrightarrow{f} D'$ be feature link in the scenario such that D' is a leaf node. Let g be the composition of feature and specialization functions along the path from the root node of α to D. Let $\beta(x)$ be the scenario such that $\alpha(x) \equiv \beta(x) \wedge D'(f(g(x))$. We made the following independence assumption in Chapter 3:

$$[D'(f(g(x))|\beta(x)]_x = [D'(f(g(x))|D(g(x)), \beta(x)]_x = [D'(f(x))|D(x)]_x$$

The scenario $\beta(x)$ consists of $D(g(x))$, the direct and indirect causes of $D(g(x))$ and their other effects. Therefore, the independence assumption means that given the cause event $D(g(x))$, the conditional probability of a causation event $D'(f(g(x)) \wedge D(g(x))$ is independent of

1. the causes of the cause event,
2. other effects of the cause event.

The first part of this assumption agrees with our intuitive notion of direct cause. An event A is said to be an indirect cause of another event B if there exists event C such that A may only influence B via C, that is, $P(B|C, A) = P(B|C)$. Since causes of the cause event are indirect causes of the causation event, they are conditionally independent of the causation event given the cause event.

For example, in our auto engine domain, `low-compression-pressure` causes `lack-of-power`. There are two causes of `low-compression-pressure`: `late-timing` and `leak-in-cylinder`. As long as they cause `low-compression-pressure` to the same degree, their effects on the `lack-of-power` are also the same.

The second part of the independence assumption is known as the Principle of Common Cause [74], which can be stated as follows:

> Let A and B be events that are approximately simultaneous and that are correlated, i.e., $P(AB) \neq P(A)P(B)$. Then the event C is a common cause of A and B if:
>
> 1. C occurs earlier than A and B;
> 2. $P(AB|C) = P(A|C)P(B|C)$;

In other words, a common cause (C) renders its effects (A and B) conditionally independent.

For example, `vomiting` and `diarrhea` are dependent. However, given their common cause `stomach-flu`, the conditional probability of `stomach-flu` causing `vomiting` and the conditional probability of `stomach-flu` causing `diarrhea` given `stomach-flu` are independent.

Sometimes, the intuition that "given the direct cause of an event, its probability is conditionally independent of the indirect causes" fails to be true. For example, generally, `unable-to-walk` causes `weak-leg`. However, if an instance of `unable-to-walk` is caused by `amputation`, then it does not cause `weak-leg`. This apparently contradicts our independence assumption because the conditional probability

$$[\texttt{weak-leg}(x)|\texttt{unable-to-walk}(x)]_x$$

is dependent on the cause of `unable-to-walk`. Fortunately, the ability to represent taxonomic hierarchy in CF-Diagram allows us to handle such cases correctly. Figure 26 shows the solution to this problem. We define a subclass of `unable-to-walk`, called `uw-c-a` (unable to walk caused by amputation) and let

$$[\texttt{weak-leg}(f(x))|\texttt{uw-c-a}(x)]_x = 0.$$

The path $\texttt{uw-c-a}\xrightarrow{f}\texttt{weak-leg}$ preempts the path $\texttt{uw-c-a} \xrightarrow{isa} \texttt{unable-to-walk} \xrightarrow{f}\texttt{weak-leg}$.

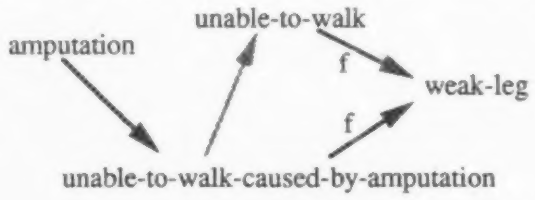


Figure 26: Specificity and causal independence

Any scenario that contains a path from `amputation` to `weak-leg` must include $\texttt{uw-c-a}\xrightarrow{f}\texttt{weak-leg}$, and therefore, has probability 0. On the other hand, if a scenario contains the causal link $\texttt{unable-to-walk} \xrightarrow{f}\texttt{weak-leg}$, then the cause of `unable-to-walk` cannot be `amputation`. In all the scenarios where the cause of `unable-to-walk` is not `amputation`, the independence assumption states that the conditional probability of `unable-to-walk` causing `weak-leg` given `unable-to-walk` is independent of the cause of `unable-to-walk`.

5.1.4. Examples from auto diagnosis

We now use a few examples to illustrate some of the characteristics of application of obvious abduction to diagnosis.

Use of attributes Consistency plays an important role in abductive reasoning. The conclusions of abductive inference (explanations) must be consistent with the knowledge base. In obvious abduction, explanations must be consistent with attribute value constraints.

For example, `misfire` causes `uneven-sound`. There are two possible causes of `misfire`: `bad-spark` and `impure-fuel`. The former is an ignition system problem and tends to happen continually. On the other hand, `impure-fuel` tends to be intermittent. Thus, its occurrences are intermittent. We use an attribute `occurrence` to encode these facts in our knowledge base:

```
(defcategory uneven-sound
  (attributes :export (occurrence)))

(defcategory misfiring
  (attributes :export (occurrence))
  (defeature cause-abnormal-sound uneven-sound
    (percolate occurrence occurrence)))

(defcategory impure-fuel
  (attributes :export (occurrence))
  (defeature cause-misfiring misfiring
    (percolate occurrence occurrence))
  (local-constraints
   (eq occurrence intermittent)))

(defcategory bad-spark
(attributes :export (occurrence))
(defeature cause-misfiring misfiring
  (percolate occurrence occurrence))
(local-constraints
 (eq occurrence continual)))
```

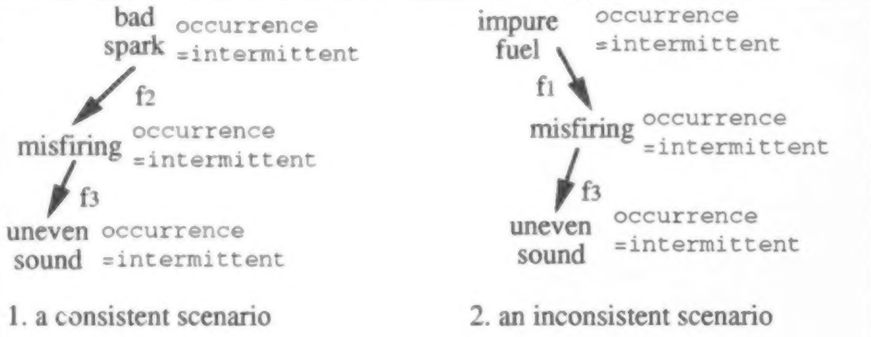
Suppose continual `uneven-sound` is observed:

$$\exists x.\texttt{uneven-sound}(x) \wedge \texttt{occurrence}(x) = \texttt{continual}$$

then the attribute value `occurrence=continual` percolates from `uneven-sound` to `misfire` and then to `bad-spark` and `impure-fuel` (see Figure 27). In Figure 27.2, the attribute value `occurrence=continual` of `impure-fuel` violates the local constraint at the node `impure-fuel`.

$$\forall x.\texttt{impure-fuel}(x) \rightarrow \texttt{occurrence}(x) = \texttt{intermittent}.$$

Therefore the scenario in Figure 27.2 is inconsistent with the knowledge base and subsequently eliminated as a candidate explanation. On the other hand, the scenario in Figure 27.1 is consistent with the knowledge base and logically implies the observations. Therefore, it is an explanation of the observation.


bad spark
occurrence =intermittent
f2
misfiring
occurrence =intermittent
f3
uneven sound
occurrence =intermittent
impure fuel
occurrence =intermittent
f1
misfiring
occurrence =intermittent
f3
uneven sound
occurrence =intermittent

1. a consistent scenario

2. an inconsistent scenario

Figure 27: Misfiring scenarios

Circular causal links Causal tendency, unlike token causation, can be circular. A simple example of cyclic causal tendency is `chicken` and `egg`. `Chicken` may cause an `egg` and an `egg` may cause a `chicken`. Token causation, on the other hand, is always acyclic. That is, a `chicken` may lead to another `chicken`, however, the latter is always a different `chicken`. The acyclicity of token causation is reflected in the acyclicity of scenarios, which has a tree structure.

Since the links in our network represent causal tendency between two types of events, there can be directed cycles in the network. In our auto engine diagnosis, `worn-piston` tends to cause `leak-in-cylinder`, which tends to cause `contaminated-oil` which again tends to cause `worn-piston`. Therefore a case of `worn-piston` may leads to a more severe case of `worn-piston`.

Faults $\cap$ Symptoms $\neq \emptyset$ Unlike many other abductive diagnostic systems, such as GSC [50], NESTOR [11], CHECK [10], the set of symptoms and the set of faults need not be disjoint. We define a fault to be any event to which a remedy can be applied, and a symptom to be any event that is observable or can be detected. Therefore, it is possible for an event to be both a fault and a symptom. For example, `loose-fan-belt` is a fault that is capable of causing other events such as `overheating` of engine. In the meantime, it is also a symptom that can easily be detected.

5.2. Plan Recognition

Since the representation of a plan hierarchy as a CF-Diagram has already been demonstrated in the examples in Chapter 3, here we only address two issues in plan recognition: specificity and recursion.

5.2.1. Specificity

Most researchers agree that specificity has something to do with the quality of explanations. However, there seem to be a confusion as to how. There are proposals such as "most specific abduction", "least specific abduction", "predicate specific abduction", "chained specific abduction". We believe that this confusion arises mostly because there are two seemingly conflicting intuitions:

1. The more specific information should supersede the less specific information. This is the same intuition as in inheritance reasoning [78].
2. The more specific an explanation is the more detailed assumptions it involves; and the more detailed an assumption is, the less likely it is true.

The most specific abduction is based on the first intuition, whereas the least specific abduction is based on the second intuition. Obvious abduction makes use of both intuitions. The apparent conflict is avoided by using these intuitions in two different places. The first intuition is used in the definition of explanations. The explanations must be maximally specific with respect to the assumptions and observations. The second intuition is used when comparing two maximally specific explanations. The more general one is preferred.

For example, suppose the observation is `make-noodle`, then the scenarios (1) and (2) in Figure 28 are explanations. Scenario (3) is not an explanation because the path

$$\texttt{make-spaghetti-marinara} \xrightarrow{\texttt{isa}} \texttt{make-noodle} \xrightarrow{f_2} \texttt{make-sauce}$$

is preempted by the path

$$\texttt{make-spaghetti-marinara} \xrightarrow{f_2} \texttt{make-marinara} \xrightarrow{\texttt{isa}} \texttt{make-sauce}.$$

Note that the specificity of an argument is relative to its antecedents. The argument in (2) is not more specific than the one in (1) because (2) used a different antecedent than (1). In inheritance reasoning, the statement "penguins don't fly" supersedes the statement "penguins, as a kind of birds, fly," but it does not preempts the statement "birds fly." Therefore, in defining maximally specific

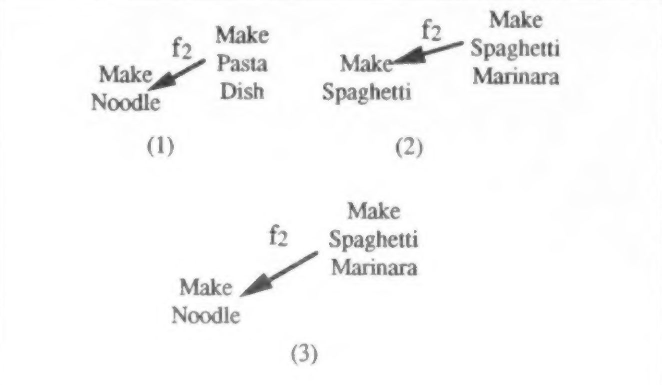


Figure 28: Specificity of explanations

explanations, we only compare the specificity of explanations that share the same set of assumptions.

Once we rule out (3) as a candidate explanation, we can compare the explanations (1) and (2). The explanation (1) is preferred because it does not make excessively detailed an assumption as (2) does.

5.2.2. Recursive plans

Recursion is a common problem solving technique. Many plans are recursive. Since the cooking world is too simple to involve any recursive plans, we will take our examples from another problem domain: **Algorithm Recognition.** In particular, we study of the recognition of sorting algorithm. The problem can be stated as follows:

Suppose we are able to observe two basic actions a program performs on a list:

1. Comparing two elements in the list. We write "cmp(i, j)=GE" to denote the observation that the program compared the i'th and the j'th element in the list and the i'th element is greater or equal to the j'th element. Similarly, LT stands for "less than."

2. Exchanging the positions of two elements in the list. We write "exch(i, j)"

to denote the observation that the program exchanged the positions of the i'th and the j'th element in the list.

Given a sequence of observations of program actions on the list, the algorithm recognition problem is to infer which algorithm the program is using.

For example, given the observations in Figure 29.a, we would like to infer that the program is using the `quickSort` algorithm in Figure 30. If the observations are as shown in Figure 29.b, we infer that the program is using the insertion sort algorithm in Figure 31.

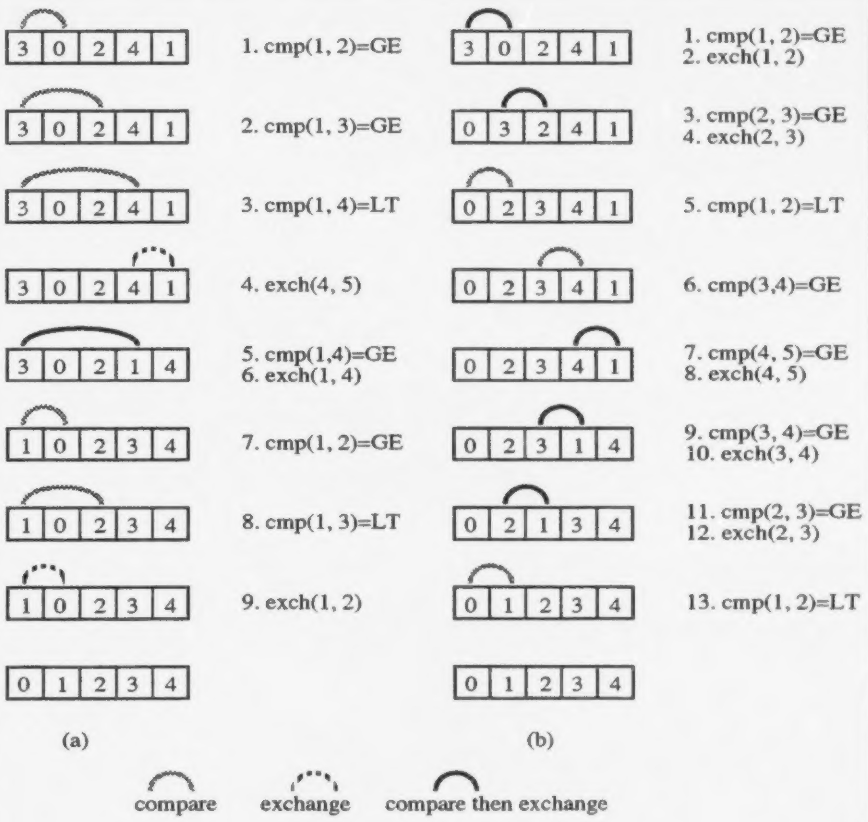

3 0 2 4 1
1. cmp(1, 2)=GE
3 0 2 4 1
2. cmp(1, 3)=GE
3 0 2 4 1
3. cmp(1, 4)=LT
3 0 2 4 1
4. exch(4, 5)
3 0 2 1 4
5. cmp(1,4)=GE
6. exch(1, 4)
1 0 2 3 4
7. cmp(1, 2)=GE
1 0 2 3 4
8. cmp(1, 3)=LT
1 0 2 3 4
9. exch(1, 2)
0 1 2 3 4
(a)
3 0 2 4 1
1. cmp(1, 2)=GE
2. exch(1, 2)
0 3 2 4 1
3. cmp(2, 3)=GE
4. exch(2, 3)
0 2 3 4 1
5. cmp(1, 2)=LT
0 2 3 4 1
6. cmp(3,4)=GE
0 2 3 4 1
7. cmp(4, 5)=GE
8. exch(4, 5)
0 2 3 1 4
9. cmp(3, 4)=GE
10. exch(3, 4)
0 2 1 3 4
11. cmp(2, 3)=GE
12. exch(2, 3)
0 1 2 3 4
13. cmp(1, 2)=LT
0 1 2 3 4
(b)
compare
exchange
compare then exchange

Figure 29: Sorting a list

Algorithm recognition problem is a novel problem proposed in this dissertation. It shares the same goal as the program recognition problem [31, 81]. Program recognition is analogous to understanding a program by reading the code. Algorithm recognition is analogous to understanding a program by experimenting with it and observe its behavious.

A possible application of algorithm recognition is to check students' assignments. For example, if the assignment requires the students to write `quickSort` algorithms, an implementation of a bubble sort will not be satisfactory even though it sorts the list correctly.

If we have only a small number of candidate algorithms, we can solve this problem by executing the candidate algorithms with the same input as the program being recognized and compare the sequences of basic actions of the candidate algorithms. If a candidate algorithm generates the same action sequence as the program being recognized, we may abduce that the program is using that algorithm.

Unfortunately, there are often a large number of possible candidates. Take `quickSort` for example, the basic scheme is:

1. Select a pivot.
2. Use the pivot to partition the list into two sub-lists such that the lower sub-list contains all the elements that are smaller or equal to the pivot and the higher sub-list contains all the elements that are larger than the pivot.
3. Sort the two sub-lists.

There are many variations to this basic scheme. Each step may be performed slightly differently.

- In selecting the pivot, we can select the first element, the middle element, the last element or the median of the first, middle and last element, and many more.
- There are many ways to partition a list as well. For example, `partition1` in Figure 30 may be replaced by `partition2` in Figure 32. Alternatively, the

subroutine `partition11` in `partition1` may be replaced by `partition12` in Figure 33.

- Finally, in the recursion step, an algorithm may use a totally different sorting algorithm to sort the sub-lists.

The possibilities of each step multiplies to generate a large number of total possibilities.

The algorithm recognition problem can be viewed as a kind of obvious abduction problem. An algorithm is a plan to accomplish some goal, which consists of a set of sub-plans. The sub-plans can further be decomposed into finer plans. Therefore, algorithms may be represented by a CF-Diagram that shows the decompositional and abstraction relationships among the components of the algorithms. For example, the algorithms in Figure 30 through Figure 33 are represented in the CF-Diagram in Figure 34, which is explained as follows:

- A plan to sort a list subsumes plans to use `quickSort` or `insertionSort`. Thus the "isa" and specialization links between sort-list and {`quickSort`, `insertionSort`}.
- In insertion sort, one first sorts the sub-list from first to the second last element and then inserts the last element into the sorted sub-list. These facts are stated in $\mathbf{S}^3$ as follows:

```
(defcategory insertion-sort
  (isa sort-list)
  (defeature sort-front insertion-sort
    (percolate low low)
    (percolate high (+1 high)))
  (defeature last-step insert-last
    (percolate low low)
    (percolate high high))
  (local-constraint
    (lt low high))
```

  The percolation constraints ensure that the recursive component sorts the sub-list from the first to the second last element.

```
void quickSort(DataType l[ ], int low, int high)
{
  int mid;
  if (low < high) {
    mid = partition1(l, low, high);
    quickSort(l, low, mid-1);
    quickSort(l, mid+1, high);
  }
}

int partition1(DataType l[ ], int low, int high)
{
  int lastSmall;
  selectPivot(l, low, high);
  lastSmall = partition11(low, l, low+1, high);
  exch(l, low, lastSmall);
  return lastSmall;
}

int partition11(int pivot, DataType l[ ], int low, int high)
{
  int cmpResult, lastSmall;
  cmpResult = cmp(l, pivot, low);
  if  (low==high) {
    if (cmpResult==GE) lastSmall = low;
      else lastSmall = low-1;
    return lastSmall;
  }
  else {
    if (cmpResult==LT) {
      exch(l, low, high);
      return partition11(pivot, l, low, high-1);
    }
    else return partition11(pivot, l, low+1, high);
  }
}
```

Figure 30: A quickSort algorithm

```
void insertionSort(DataType l[ ], int low, int high)
{
  if (low==high) return;
  insertionSort(l, low, high-1);
  insertLast(l, low, high);
}

void insertLast(DataType l[ ], int low, int high)
{
  if (low==high) return;
  if (cmp(l, high-1, high)==GE) {
    exch(l, high-1, high);
    insertLast(l, low, high-1);
  }
}
```

Figure 31: An insertion sort algorithm

```
int partition2(DataType l[ ], int low, int high)
{
  if (low==high) return low;
  if (cmp(l, low, low+1)==LT) {
    exch(l, low+1, high);
    return partition2(l, low, high-1);
  }
  else {
    exch(l, low, low+1);
    return partition2(l, low+1, high);
  }
}
```

Figure 32: An alternative for `partition1`

```
int partition12(int pivot, DataType l[ ], int low, int high)
{
  int newLow, newHigh;
  newLow = increaseLow(pivot, l, low, high);
  newHigh = decreaseHigh(pivot, l, low, high);
  if  (newLow <= newHigh) {
    exch(l, newLow, newHigh);
    return partition12(pivot, l, newLow+1, newHigh-1);
  }
  else return newHigh;
}

int increaseLow(int pivot, DataType l[ ], int low, int high)
{
  while (low <= high and cmp(l, pivot, low)==GE)
    low = low+1;
  return low;
}

int decreaseHigh(int pivot, DataType l[ ], int low, int high)
{
  while (low <= high and cmp(l, pivot, high)==LT)
    high = high-1;
  return high;
}
```

Figure 33: An alternative for `partition11`

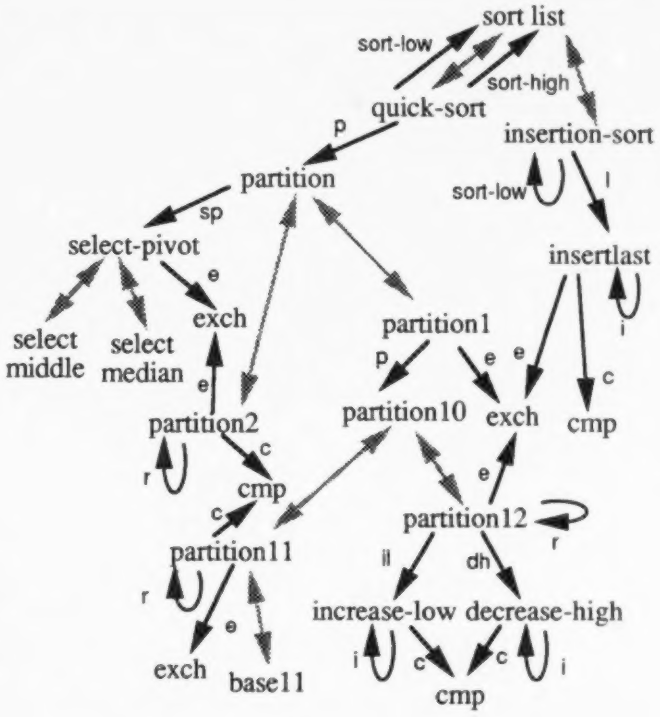

sort list
sort-low
sort-high
quick-sort
p
insertion-sort
partition
sort-low
l
sp
select-pivot
e
insertlast
exch
select middle
select median
e
partition1
i
p
e
e
c
partition2
c
r
partition10
exch
cmp
cmp
e
c
partition12
r
partition11
il
dh
r
e
increase-low
decrease-high
exch
base11
i
c
c
i
cmp

Figure 34: A CF-Diagram of sorting algorithms

- `quickSort` has two feature links to sort-list because `quickSort` has two sort-list components. The reason that these two links are pointed to sort-list instead of `quickSort` is that it may be desirable to employ other sorting algorithms (e.g., `insertionSort`) to sort the sub-lists when the sub-lists are short.
- `partition1` and `partition2` are two methods of partitioning a list. Therefore, they are subclasses of `partition`. Both of them has a select-pivot component. This component is only defined once as a component of `partition` and is inherited by the subclasses.
- `partition1` first use `partition10` to partition the sub-list from the second to the last element. `partition10` therefore may assume that the pivot is

outside the list being partitioned and returns position of the last element that is less than or equal to the pivot, called `lastSmall`. Then `partition10` swap the elements at pivot and `lastSmall`.

- Figure 34 shows two implementations of `partition10`:

  `partition11` and `partition12`

  which are shown in Figure 30 and Figure 33, respectively.
- `base11` is a special case of `partition11`, where the list being partitioned contains only one element.
- The multiple instances of `cmp` nodes in Figure 34 represent a single `cmp` node to avoid too many overlapping links. Same can be said about `exch`.
- A complete specification of the CF-Diagram in $\mathbf{S}^3$ can be found in the Appendix.

Note that the $\mathbf{S}^3$ representation has abstracted away some of the syntactic variations of the algorithms. For example, the insertion sort algorithm in Figure 31 used recursive procedure calls. The same algorithm can also be written with "while" loops, without changing its $\mathbf{S}^3$ representation.

Given the above CF-Diagram representation of sorting algorithms, and a sequence of basic actions, our abductive reasoning algorithm is able to construct a scenario that explains the sequence of observations as the sequence of actions during the execution of an algorithm.

For example, suppose the observations are those in Figure 35. Each of the observations results in an original item that is fed into the CF-Diagram, which then initiates a message passing process. When all the original items have been fed into the network and the message passing process stops, the scenarios that explain the observations can be retrieved from the network by tracing the origins of the items whose O-component contain all the observations. Figure 36 shows such a scenario. The items 1, 2, 3, 4, and 5 are original items representing the observations. The items *a* through *l* are generated during the message passing process. This scenario tells us that the sorting algorithm is `quickSort` where `partition11` is used to partition the list.

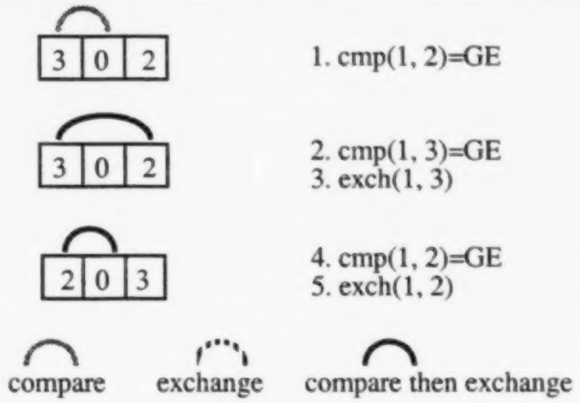

3 0 2
1. cmp(1, 2)=GE
3 0 2
2. cmp(1, 3)=GE
3. exch(1, 3)
2 0 3
4. cmp(1, 2)=GE
5. exch(1, 2)
compare
exchange
compare then exchange

Figure 35: Observations about a sorting program

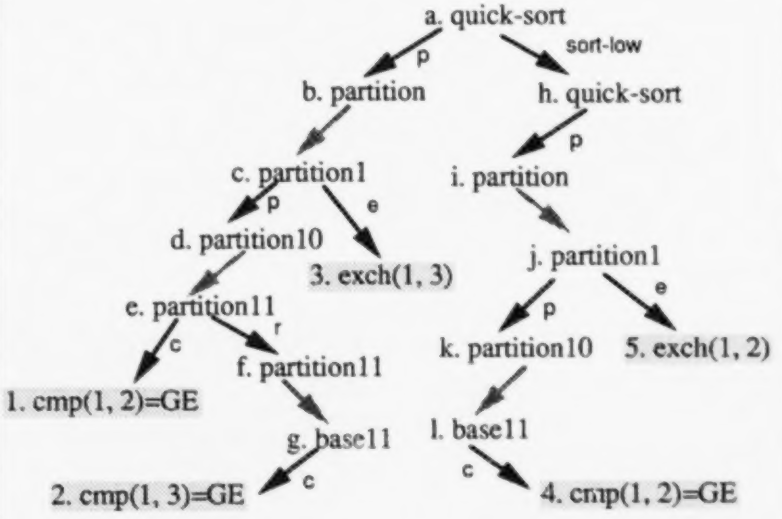

a. quick-sort
p
sort-low
b. partition
h. quick-sort
p
c. partition1
i. partition
p
e
d. partition10
3. exch(1, 3)
j. partition1
e. partition11
r
p
e
c
f. partition11
k. partition10
5. exch(1, 2)
1. cmp(1, 2)=GE
g. base11
l. base11
c
c
2. cmp(1, 3)=GE
4. cmp(1, 2)=GE

Figure 36: The explanation of actions in Figure 35

The items in the trace in Figure 36 are shown in Table 5.1. The process by which the scenario in Figure 36 is obtained can be briefly described as follows:

Table 5.1: Items in the scenario in Figure 36

item	location			

1	`cmp`	<{1},	`left = 1`
			`right = 2`
			`cmpResult = GE`
				,{}>

2	`cmp`	<{2},	`left = 1`
			`right = 3`
			`cmpResult = GE`
				,{}>

3	`exch`	<{3},	`left = 1` `right = 3`	,{}>

4	`cmp`	<{4},	`left = 1`
			`right = 2`
			`cmpResult = GE`
				,{}>

5	`exch`	<{5},	`left = 1` `right = 2`	,{}>

a	`quick-sort`	<{1,2,3,4,5},	`low = 1`
			`high = 3`
			`mid = 3`
				,{*p*, *sort-low*}>

c	`partition1`	<{1,2,3},	`low = 1`
			`high = 3`
			`mid = 3`
				,{*p*, *e*}>

e	`partition11`	<{1,2},	`pivot = 1`
			`low = 2`
			`result = GE`
			`high = 3`
			`lastSmall = 3`
			`subLow = 3`
			`subHigh = 3`
				,{*c*, *r*}>

g	`base11`	<{2},	[`pivot = 1`
			`low = 3`
			`result = GE`
			`high = 3`
			`lastSmall = 3`]
				,{*c*}>

h	`quick-sort`	<{4,5},	`low = 1`
			`high = 2`
			`mid = 2`
				,{*p*}>

j	`partition1`	<{4,5},	`low = 1`
			`high = 2`
			`mid = 2`
				,{*p*, *e*}>

l	`base11`	<{4},	[`pivot = 1`
			`low = 2`
			`result = GE`
			`high = 2`
			`lastSmall = 2`]
				,{*c*}>

1. The first observation `1.cmp(1, 2)=GE` generates the item 1 in Table 5.1 at the node `cmp` which sends the item to its parents: `partition2, partition11, insertlast, increase-low, decrease-high`. The message is also forwarded to `base11` by its superclass `partition11`.

2. The second observation is represented by the item 2 in Table 5.1. When the node `cmp` receives the item, it is sent to `base11`. The percolation constraint forces the attribute values of `pivot, low` and `result` at the node `base11` to be the values of `left, right` and `cmpResult` respectively. The local constraint at the node `base11` then forces the attributes `high` and `lastSmall` to be the value of `low`. Therefore, the item g in Table 5.1 is created at the node `base11`.

3. The item g is sent across the specialization link to the node `partition11` and then to `partition11` again via the recursive feature r. An item is created at the node `partition11` whose attribute values `subLow, subHigh, pivot, lastSmall` are determined by the percolation constraints associated with feature r. The item is unified with another item received from `cmp` which represents the first observation. The unification of the two items is the item e at the node `partition11`.

4. The item e is sent to `partition1` via a specialization link and a feature link p. It is combined with the item 3 from the node `exch` and their unification, item c is sent to the node `quick-sort` via a specialization link and the feature link p.

5. The items 4 and 5 represent observations 4 and 5 respectively. They generate the item h at the node `quick-sort` after propagation and combination. The item h is sent to the node `quick-sort` again via the recursive link r and combines with the item c that came from the node partition to form the

node a, whose O-component contains all the observations. The scenario in Figure 36 is the trace of the item a.

5.3. Parsing as abduction

This section is concerned with the transformation of a grammar into a network and the complexity analysis of our abductive reasoning algorithm in CFG parsing. We also generalize CFG parsing by augmenting grammatical categories with attributes and constraints over the attribute values so that linguistic phenomena such as agreement can be dealt with.

5.3.1. Translation of CFG into $\mathbf{S}^3$

Context-Free Grammars are usually represented by a set of rules. In Section 2.3.1, we showed that the rules can be represented by a grammar network. The grammar network is also a kind of CF-Diagram which can be specified in $\mathbf{S}^3$ as follows:

Each non-terminal symbol n is a category:

```
(defcategory n)
```

The rules are also represented by categories. Given a CFG rule :

$$R : m \longrightarrow d_0 \ldots d_k$$

it is translated into the following **`defcategory`**:

```
(defcategory R
  (isa m)
  (defeature f_0 d_0)

  ... ...

  (defeature f_k d_k))
```

That is, R is a subclass of m, for each category d_i on the right hand side of the rule, there is a feature link from R to d_i.

It can be seen that the `AndLinks` in Section 2.3.1 are feature links. The `OrLinks` are specialization links. In a CF-Diagram representing a CFG, the "isa" links are redundant because no category inherits features from its superclasses. All the features in the CF-Diagram are from the rule nodes to the non-terminal nodes and the rule nodes do not have any subclasses. This explains why the grammar network in Figure 7 does not contain any "isa" links.

In parsing, the observations are a sequence (a totally ordered set) of words, rather than simply a set. Correspondingly, the explanations consist of sub-explanations that explain sub-sequences of the observations. There is also a total order among the categories on the right hand side of a CFG rule.

These ordering constraints are enforced by the additional restriction on the unification of items. In the algorithm in Chapter 4, two items $i \equiv <O, V, F>$ and $i' \equiv <O', V', F'>$ are compatible if O and O' are disjoint, and F and F' are disjoint. They are unifiable if they are compatible and V and V' are unifiable. Let $w_0, w_1, \ldots, w_n$ be the words in the input sentence and $f_0, \ldots, f_k$ be the feature links from R to $d_0, \ldots, d_k$. To parse CFG sentences, we redefined the generic method `compatibleItems` so that two items are compatible iff the following are true:

1. O and O' are disjoint;
2. F and F' are disjoint;
3. $O \cup O' = \{w_i, w_{i+1}, \ldots, w_{j-1}, w_j\}$ for some $i < j$.
4. $F \cup F' = \{f_0, f_1, \ldots, f_{i-1}, f_i\}$ for some $0 \leq i \leq k$.

For example, $<\{w_1, w_2\}, [], \{f_0, f_1\}>$ is unifiable with $<\{w_3, w_4\}, [], \{f_2\}>$. On the other hand, if

$$O = \{w_1, w_2\}, O' = \{w_0, w_3\} \quad \text{or} \quad F = \{f_1, f_2\}, F' = \{f_3\}$$

then $<O, [], F>$ and $<O', [], F'>$ are not unifiable.

The constraints on item unification ensure that the surface string of m is the concatenation of the surface strings of d_i's ($i = 0, 1, \ldots, k$). The experimental results of an implementation is shown in Appendix B.

5.3.2. Complexity analysis of CFG parsing

In this subsection, we show that the complexity of CFG parsing with obvious abduction is

$$O(|G|n^3)$$

where $|G|$ is the size of the grammar (the total number of occurrences of non-terminal symbols in the rules in G) and n is the number of words in the input sentence. This is a significant improvement over many previous CFG parsing algorithms, such as Earley's parser, and chart parsers, which have the complexity $O(|G|^2n^3)$ [3, p.198], because natural language grammars tend to have a large number of rules but relatively small number of words in a sentence.

The complexity of the algorithm is the sum of the complexity of message passing and local processing at each node. Since the items that are sent across the same link must have different O-component and identical F-component, the maximum number of such items is bounded by $O(n^2)$. Therefore, the complexity of message passing part is bounded by $O(|G|n^2)$.

When a node receives an item, it tries to unify the newly received item with compatible items at the node. The complexity of local processing is bounded by the number of compatible items the nodes *attempt* to unify. Suppose the newly received item is:

$$<\{w_j, w_{j+1}, \ldots, w_l\}, [], \{f_m\}>$$

then the compatible items must be in the form:

$$<\{w_i, w_{i+1}, \ldots, w_{j-1}\}, [], \{f_0, f_1, \ldots, f_{m-1}\}>$$

where $i < j \leq n$. Their unification is

$$<\{w_i, w_{i+1}, \ldots, w_l\}, [], \{f_0, f_1, \ldots, f_m\}>$$

Thus, there are at most $n-1$ compatible items, which implies that there are at most $n-1$ unifications attempted for each item received by a node. Since the nodes in the network receive at most $(O(|G| \times n^2)$ items when parsing a sentence with n words, the complexity of local processing is bounded by $O(|G| \times n^3)$. Therefore, the complexity of CFG parsing is:

$$O(|G|n^2 + |G| \times n^3) = O(|G|n^3)$$

5.3.3. ID/LP parsing

A CFG rule specifies not only the immediate dominance (ID) relation between the left-hand-side symbol over the right-hand-side symbols, but also the linear precedence (LP) relationships among the right-hand-side symbols. In many natural languages, the order of the right-hand-side symbols in a rule is often very flexible. Therefore, a more compact representation of the grammar can be obtained by separating the ID and LP specification. An ID-rule is in the form:

$$m_i \xrightarrow{\text{ID}} d_{i_0}, \ldots, d_{i_{k_i}}$$

which leaves the ordering of the right-hand-side symbol d_{i_0}, ..., $d_{i_{k_i}}$ unspecified.

One obvious way to parse grammars with ID-rules is to expand each ID-rule into several CFG rules and apply a CFG parser on the object grammar. In contrast, direct ID/LP parsing uses ID-rules directly. It has been shown in [3, p.194] that, although the ID/LP parsing is NP-complete, it is still more efficient than the alternative of parsing the expanded "object grammar."

Direct ID/LP parsing can also be handled by obvious abduction. ID-rules are translated into a CF-Diagram similarly to CFG rules. Given an ID-rule:

$$R_i : \quad m_i \xrightarrow{\text{ID}} d_{i_0}, \ldots, d_{i_k}$$

there are $k+1$ feature links from R_i to d_{i_j} and an "isa" link from R_i to m_i. The conditions for item unification in ID/LP parsing is the same as in CFG parsing, except that the forth condition becomes that the $F \cup F'$ satisfies LP constraints.

When an item

$$\{w_j, w_{j+1}, \ldots, w_l\}, [], \{f_m\}.$$

arrives at a node, the compatible items must be in the form

$$<\{w_i, w_{i+1}, \ldots, w_l\}, [], F>$$

where $F \cap \{f_m\} = \emptyset$. There are j possible values for i and 2^d possible values for F, where d is the maximum non-terminal symbols on the right hand side of an ID-rule. Therefore, there are at most $O(2^d n)$ compatibles items. The complexity of the direct ID-rule parsing is $O(2^d n^3 |G|)$.

5.3.4. Use of attributes

The grammar in Figure 5 will admit ungrammatical strings such as

* John are sick.

The problem here is what linguists call "agreement". Many contemporary linguistic theories, such as Generalized Phrase Structure Grammar (GPSG) [17] and Lexical-Functional Grammar (LFG), employ attributes (or features) to cope with such phenomenon in natural language. The nonterminal categories such as NP and V are augmented with or replaced by as set of attributes. For example, an NP object may have an attribute `case` which takes `accusative` or `nominative` as its value. The attribute values are governed by a set of constraint principles so that in admissible parse trees, the constituents must satisfy the requirements such as subject–verb agreement.

The percolation and local constraints in $\mathbf{S}^3$ can also be used to enforce constraints such as subject–verb agreement. For example,

```
(defcategory R_S1
  (isa S)
  (attributes
    :export (low high)
    :local (number mid))
  (defeature subject NP
    (percolate low low)
    (percolate high mid)
    (percolate number num))
  (defeature predicate VP
    (percolate low mid)
    (percolate high high)
    (percolate number num)))
```

The percolation constraints ensure that the `number` attribute of the verb phrase equals to that of the subject. If the input sentence is "*John are sick," then the `number` attribute of the NP "John" is `singular` and the `number` attribute of the VP "are sick" is `plural`. The NP and the VP do not form a sentence S because the NP forces the `number` attribute of S to be `singular`, whereas the VP forces the value to be `plural`.

Chapter 6

Related work

This chapter summarize other abductive reasoning approaches and compares them to obvious abduction. Our purpose is to clearly identify strengths and the weaknesses of each approach and the major differences/similarities between the approaches.

6.1. Domain independent abduction

6.1.1. Theorist

Theorist [56] is a hypothetical reasoning system based on an ordinary first order clausal theorem prover that distinguishes facts from hypotheses. Theorist consists of:

1. a set F of *facts*, representing things that are known to be true in the domain;
2. a set H of *possible hypotheses* that may be assumed to be true.

In most implementations of Theorist, the representation language is full first-order clausal logic. F is a consistent set of closed formulae and H is a set of ground literals, usually expressed as a set of sentence schemas which represent the set of all possible instantiating ground literals. F and H are specified as follows:

fact *Clause* means that *Clause* $\in F$.
hypothesis *Name* means that *Name* $\in H$.

hypothesis *Name*: *Clause* is a shorthand for

fact *Name* → *Clause*
hypothesis *Name*

Given a goal sentence G, reasoning in the Theorist framework involves finding consistent subset E of the possible hypotheses H, which, together with the *facts* F, logically entails G. E is called the explanation of G.

Definition 6.1 (Explanation in Theorist) $E \subseteq H$ is an explanation of G if

1. $F \cup E \models G$, and
2. $F \cup E$ is consistent.

This specification is illustrated in Figure 37.

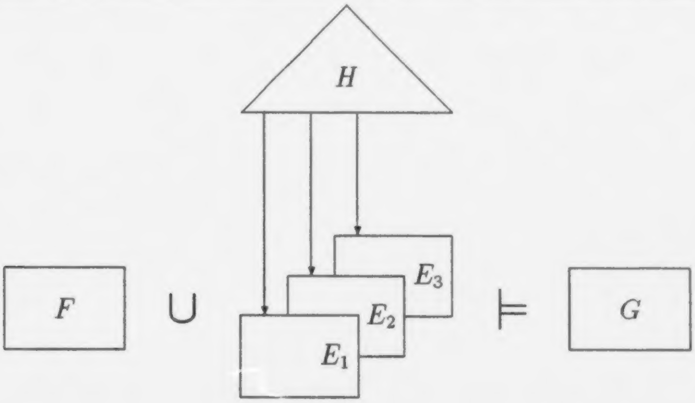


Figure 37: Theorist reasoning architecture

To illustrate the basic Theorist framework, the well-known birds fly example is represented as follows [23]:

fact bird(tweety).
fact bird(X) ← penguin(X).
fact ¬flies(X) ← penguin(X).
hypothesis bf(X): flies(X) ← bird(X).

This encodes the knowledge that Tweety is a bird and all penguins are birds and penguins don't fly. As well, if something is a bird then it can be assumed to fly provided the assumption is consistent. From this, G_1 = {flies(tweety)} can be explained with the theory E_1 = { bf(tweety)} formed by instantiating the hypothesis bf(X) with X = tweety. Therefore we can explain that Tweety flies because Tweety is a bird and birds can be assumed to fly. Should we later learn that

fact penguin(tweety).

then G_1 can no longer be explained as E_1 is inconsistent with the facts. However G_2 = {¬ flies(tweety)} can be explained with the empty theory E_2= {}, i.e., Tweety doesn't fly follows deductively from the facts.

In general, there may be multiple theories that explain the observations. These correspond to multiple minimal models in circumscription [42, 38] and multiple extensions in default logic [60, 55]. Theorist has been extended to include both general and domain dependent methods of pruning explanations [66, 19, 24].

6.1.2. Levesque

Similar to Theorist, in the knowledge-level account of abduction proposed by Levesque [36], abduction is also defined to be the search for a set of hypotheses that entail the observations. However, the entailment is characterized in terms of belief model, which coincides with logical entailment when belief is closed under ordinary logical consequence. Different forms of abduction can be studied by varying the underlying notion of belief.

Let $\mathcal{L}$ be a standard propositional language, where falsity is denoted by $\Box$. Belief is expressed in $\mathcal{L}^*$, which is structured like $\mathcal{L}$ except that all of its atomic sentences are of the form $\mathbf{B}_\lambda\alpha$, where λ is the type of the belief, α is a sentence of $\mathcal{L}$. An *epistemic state* is a specification of the truth value of atomic sentences in $\mathcal{L}^*$.

An explanation w.r.t. an epistemic state e for a type of belief λ is defined as follows:

Definition 6.2 A sentence $\alpha \in \mathcal{L}$ explains a sentence $\beta \in \mathcal{L}$ with respect to e iff

$$e \models [\mathbf{B}_\lambda(\alpha \rightarrow \beta) \wedge \neg\mathbf{B}_\lambda\neg\alpha]$$

and is denoted by α *expl*$_\lambda$ β w.r.t. e.

Instead of explicitly listing all the potential hypotheses, Levesque uses a syntactic criterion to define the simplest explanations. Abduction is defined as the process of finding the disjunction of simplest explanations for a sentence β (written $\mathrm{EXPLAIN}_\lambda[e, \beta]$). Simplicity here is interpreted as follows:

Definition 6.3 The literals of α, LITS(α), is defined by:

$\mathrm{LITS}(\square) = \emptyset$;
$\mathrm{LITS}(p) = \{p\}$;
$\mathrm{LITS}(\neg\alpha) = \{\neg m \mid m \in \mathrm{LITS}(\alpha)\}$;
$\mathrm{LITS}(\alpha \wedge \beta) = \mathrm{LITS}(\alpha \vee \beta) = \mathrm{LITS}(\alpha) \cup \mathrm{LITS}(\beta)$;

Definition 6.4 α is *simpler* than β (written $\alpha \prec \beta$) iff $\mathrm{LITS}(\alpha) \subset \mathrm{LITS}(\beta)$;

Therefore, simpler means "containing fewer propositional letters," but keeping track of their polarity.

Definition 6.5 α min-expl$_\lambda$ β w.r.t. e iff α *expl*$_\lambda$ β and there is no $\alpha' \prec \alpha$ such that α' *expl*$_\lambda$ β w.r.t. e.

Definition 6.6 $\mathrm{EXPLAIN}_\lambda[e, \beta] = \bigvee_{\alpha \text{ min-expl}_\lambda\, \beta \text{ w.r.t. } e}\, \alpha$

Therefore, given a knowledge base KB (a set of sentences), a sentence β and a function R_λ that maps (finite) sets of sentences into epistemic states, the abductive reasoning is defined as the procedure of finding the disjunction of the explanations: $\mathrm{EXPLAIN}_\lambda[R_\lambda(\mathrm{KB}), \beta]$.

To justify the definition of simplest explanation, Levesque showed that the simplest explanations coincide with the ATMS [14, 63] specification when KB is a set of Horn clauses, β is a propositional letter and belief is closed under logical consequence [35].

6.1.3. TACITUS

In the TACITUS (The Abductive Commonsense Inference Text Understanding System) Project at SRI, Hobbs *et al.* developed a scheme for weighted abduction[27]. The knowledge base consists of a set of Horn clauses. Each antecedent of a Horn clause is assigned a numerical weight. For example, suppose

$$Q \leftarrow P_1^{w_1}, \ldots, P_m^{w_m}$$

is a rule in the knowledge base, then using this rule to prove Q results in the assignment of weight $w_i \times c$ to P_i, where c is the cost of Q.

Similar to Theorist, observations are a set of literals and an explanation is a proof of the observations. Each literal in the observation is assigned a cost. When a literal is proved with a rule, the cost of the literal is propagated to the antecedents of the rule. The antecedents may themselves by proved by other rules and the costs will propagate accordingly. The cost of an explanation is the total cost of the assumptions used in the proof. The abduction problem is that of finding the minimum-cost explanation.

6.1.4. Ng & Mooney

Explanation ranking in the approaches reviewed up to now is based on Occam's Razor, where simplicity is interpreted as minimal subset of explanations or minimum cost. Ng & Mooney [44] claim that "Occam's Razor is not sharp enough." They believe that explanatory coherence, rather than simplicity, should be the primary consideration. In their proposal, domain knowledge is encoded in Horn clause axioms. Observations are a conjunction of positive literals. An explanation is a proof graph such that each node in the proof graph corresponds to a distinct literal used in the explanations and for each backward chaining rule

$$C \leftarrow A_1, \ldots, A_n$$

used in the explanation, there are n directed links $A_i \rightarrow C$ in the proof graph.

The coherence measure is:

$$\frac{\sum_{1 \leq i \leq j \leq l} N_{i,j}}{NC_l^2}$$

where l is the total number of observations;

N is the total number of nodes in the proof graph;

$C_l^2 = l(l-1)/2$;

$N_{i,j}$ is the number of distinct nodes n_k in the proof graph such that there is a directed path from n_k to n_i and from n_k to n_j, where n_i and n_j are observations.

The best explanation of the observations is a proof of the observations that maximize the coherence measure.

6.1.5. Thagard

The Explanatory Coherence Theory (ECT) proposed by Thagard [76] also use coherence as the criterion for ranking explanations. Unlike Ng & Mooney's measure of coherence, which is defined by an algorithmic function, Thagard's measure of coherence is computed by a connectionist network. For each set of competing explanations, he set up a network where the nodes are propositions in the explanations. The links between the network are either excitory or inhibitory. If the two propositions cohere according to the principles he identified, then there is an excitory link between them. For example,

> If $P_1, \ldots, P_m$ explains Q, then: each pair of propositions among $P_1, \ldots, P_m$, and Q cohere.

If two propositions are mutually exclusive, then there is an inhibitory link between them. The overall coherence of the network is the function:

$$\sum_i \sum_j w_{ij} a_i a_j$$

w_{ij} is the weight of the link between proposition i and proposition j. a_i is the activation level of proposition i.

6.1.6. Kautz

In his dissertation [34], Kautz presented a formal theory of plan recognition. The knowledge about plans is organized in an event hierarchy H_E. The event hierarchy consists of a collection of restricted-form first-order axioms that define the abstraction and decomposition relationships between various types of events. The axioms may be viewed as a logical encoding of a semantic network. The nodes in the network are unary predicates representing event types. The links in the semantic network fall into two categories: role-functions, which lead from plans to their components, and "isa" links. An `End` event is one that is not component of another event. Figure 38 shows the event hierarchy of Hunting and Robbing.

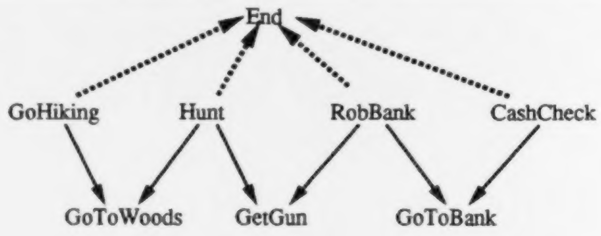


Figure 38: Hunt/Rob Hierarchy

In Kautz's theory, an explanation is a model of H_E that is consistent with the observations. Since there are too many such models, a technique known as *model minimization* is used to select a suitable subset, called *covering models.* While it would be unwise to arbitrarily adopt a particular covering model, it is reasonable to conclude whatever propositions hold in *all* covering models. Such propositions are *c-entailed* by the observation.

The model minimization technique employed by Kautz is similar to that of circumscription [42]. In fact, Kautz's minimization can be define as a series of circumscriptions [34, Theorem 3.17].

When several events are observed, c-entailment alone may not be enough to infer the agent's goal. For example, in Figure 38, suppose that {`GetGun`, `GoToBank`} is observed. This set of observations does not c-entail an instance of `RobBank`; the model containing an instance of `Hunt` and an instance of `CashCheck` provides an example. Still stronger assumptions are employed to minimize the number of `End` events in the covering model, resulting in minimum covering models. In the above example, the only minimum covering model is the model that contains an instance of `RobBank`. Therefore, `RobBank` can be concluded from the observations.

6.1.7. PCT

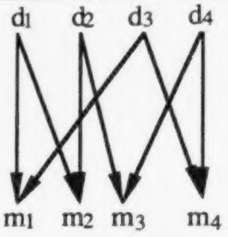
PCT (Parsimonious Covering Theory) evolved from the research in abductive diagnosis. In PCT, the domain consists of two disjoint sets of propositions:

1. a set of disorders D;
2. a set of manifestations M.

A causal relation is defined to be a function *effects* that maps a disorder to the set of manifestations it causes, i.e.,

$$\mathit{effects} : D \mapsto 2^M.$$

For example, $\mathit{effects}(d_1) = \{m_1, m_2\}$ means that the disorder d_1 is capable of causing manifestations m_1 and m_2. This causal relation can be represented as a bipartite graph where the links lead from disorders to the manifestations they are capable of causing. An example is shown in Figure 39. Given a set of manifestations $M^+ \subset M$, the abduction problem is to find the minimal subset of disorders (w.r.t. set inclusion) that covers (or, is capable of causing) the observed manifestations. For example, suppose the manifestations $\{m_1, m_2\}$ are observed, the explanations are $\{d_1\}$ and $\{d_2, d_3\}$. On the other hand, $\{d_1, d_2\}$ is not an explanation because d_2 is redundant.



$d_1 \longrightarrow \{m_1, m_2\}$

$d_2 \longrightarrow \{m_2, m_3\}$

$d_3 \longrightarrow \{m_1, m_4\}$

$d_4 \longrightarrow \{m_3, m_4\}$

Figure 39: Parsimonious covering theory

The idea of covering has long been implicit in many diagnostic systems, e.g., INTERNIST [58]. The contribution of set covering model is its clarity in the formulation of the problem, the use of provably correct algorithm for finding the minimal coverings and the principled treatment of multiple simultaneous disorders.

Peng and Reggia have also investigated the use of multi-layered network to model chaining in abduction. There are several undesirable aspects in their approach. First, the division of layers is not natural. These partition into layers is solely for the sake of using covering model. Human diagnosticians may not perceive such layering. Secondly, the hypotheses in an explanation must come from the same layer. This may be awkward in many cases. Finally, although the covering relation is transitive, irredundant covers are not transitive. For example: in Figure 40, $\{h_5, h_7\}$ is an irredundant cover of $\{h_1, h_2, h_3, h_4\}$ and $\{h_9, h_{10}\}$ is an irredundant cover of $\{h_5, h_7\}$. However, $\{h_9, h_{10}\}$ is not an irredundant cover of $\{h_1, h_2, h_3, h_4\}$.

6.1.8. Bylander *et al.*

In PCT, the effects of a set of disorders is the union of the effects of each individual disorder. To deal with more complicated multi-cause interaction, Bylander *et al.* presented a more general formulation of the set covering theory of abduction [5], where the causal relation is a function from the power set of disorders to the power

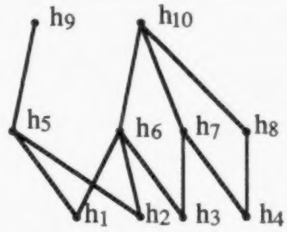
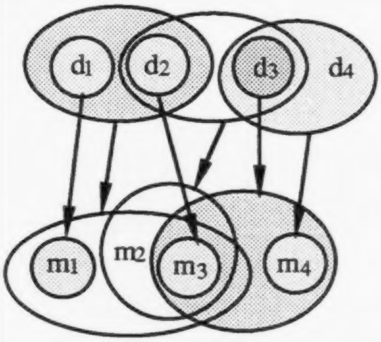


Figure 40: Parsimonious covering theory

set of manifestations:

$$\mathit{effects} : 2^D \mapsto 2^M.$$

In the example shown in Figure 41, the presence of d_1 and d_2 together causes $\{m_1, m_2, m_3\}$. However, m_2 is not caused by d_1 or d_2 alone. On the other hand, d_3 alone may cause m_3 and m_4. However, in the presense of both d_3 and d_4, d_3's effect on m_3 is canceled.



$$\{d_1\} \longrightarrow \{m_1\}$$
$$\{d_2\} \longrightarrow \{m_3\}$$
$$\{d_1, d_2\} \longrightarrow \{m_1, m_2, m_3\}$$
$$\{d_3\} \longrightarrow \{m_3, m_4\}$$
$$\{d_2, d_3\} \longrightarrow \{m_2, m_3\}$$
$$\{d_3, d_4\} \longrightarrow \{m_3, m_4\}$$

Figure 41: Set covering model of abduction

In a sense, PCT is a special case of the abduction scheme in [5], known as the "Independent Abduction." However, Peng and Reggia [51] have also investigated probabilistic reasoning in PCT and the use of generators to represent multiple diagnoses in a more compact form.

6.1.9. Bayesian Networks

Bayesian networks [47] are directed acyclic graphs where each node represents a random variable, or an uncertain quantity, which can take on two or more possible values. The network encodes the independence relationships among the random variables.

> "The advantage of network representation is that it allows people to express directly the fundamental qualitative relationship of *direct dependency*. The network then displays a consistent set of additional direct and indirect dependencies and preserves it as a stable part of the model, independent of the numerical estimates. [47, p.51]"

A **partial assignment** of the random variables in the network is an assignment of a subset of the variables in the network. A **complete assignment** is an assignment of all the variables in the network. A complete assignment is said to be consistent with a partial assignment if it assigns the same values to the variables that are assigned in the partial assignment.

Abductive reasoning in Bayesian networks takes the form of **belief revision.** The observations are represented by a partial assignment

$$e \equiv \{I_1 = i_1, I_2 = i_2, \ldots, I_n = i_n\}$$

where $I_1, I_2, \ldots, I_n$ are random variables in the network. The purpose of the belief revision is to compute the most probable complete assignment that is consistent with the observation. The complete assignment thus serves as an explanation for the observations. Note that the most probable explanation cannot be obtained simply by choosing the most probable value for each variable separately. Pearl [47, p.246] showed that sometimes this may even yield the least probable explanation.

More precisely, let W stand for the set of all variables in the Bayesian network. Let w be a complete assignment of W that is consistent with e, i.e., w assigns the value of $I_1, I_2, \ldots, I_n$ to $i_1, i_2, \ldots, i_n$ respectively. The belief revision procedure computes:

$$P(w^*) = max_w P(w|e)$$

The task of finding w^* is accomplished by computing the function

$$BEL^*_X(x) = max_{w'_X} P(x, w'_X|e)$$

at each node X, where w'_X is an assignment to variables in W other than X. $BEL^*_X(x)$ stands for the probability of the most probable instantiation of variables that is consistent with e and $X = x$.

A shortcoming of the formulation of abduction is that an explanation involves assignments to all the variables in the Bayesian, even though most of the variables are at most remotely related to the observations. To overcome this problem, Shimony [68] proposed that an explanation consist only of the varible assignments that are relevant to the observations.

The nodes in Bayesian networks represent a set of exhaustive and mutually exclusive propositions. When there are multiple instances of the same type event, a Bayesian network has to use different nodes to represent each one of them. In many domains, such as story understanding, it is impossible to know *a priori* all the individual events. Therefore, one has to construct a Bayesian network for each set of input [21, 30]. For example, in [21], a Bayesian network is constructed for every input story by a deductive database. The deductive database contains rules that are fired as each word of the story is entered. The rule then searches the database and add nodes and links in the Bayesian network. When a Bayesian network has been constructed, the usual belief propagation procedure is used to obtain the most likely interpretation. The following rule is an example in [21]:

> **Rule 2**
>
> ```
> (-> (word-inst ?i ?word) :label ?A
>     (-><- (word-sense ?word ?frame ?prob)
>           (inst ?i ?frame) :label ?C)
>     :prob ((?C -> ?A) ((t|t) ?prob) ((t|f) (/ :p 100))))
> ```
>
> This rule may be read as follows: If a node is added to the network describing a new word-token (an instance of a word), and if there is a statement in the database specifying that one of the senses of this word is `?frame`, add a node for an instance of the type `?frame`. Draw

> an arc from this newly-added node (`?C`), to the `word-inst` node (`?A`).
> ...

Therefore, in [21], the Bayesian network is only used to choose the most probable interpretation after a Bayesian network representation of all possible interpretations has been constructed. The disadvantage of this approach is that the rules used to generate the Bayesian networks are non-modular and may be difficult to verify.

In contrast, the nodes in a CF-Diagram represent predicates. A scenario may contain multiple instances of the same node. Therefore, the domain knowledge can be represented by a single network and we use the abductive inference algorithm to retrieve the most probable scenario from the network.

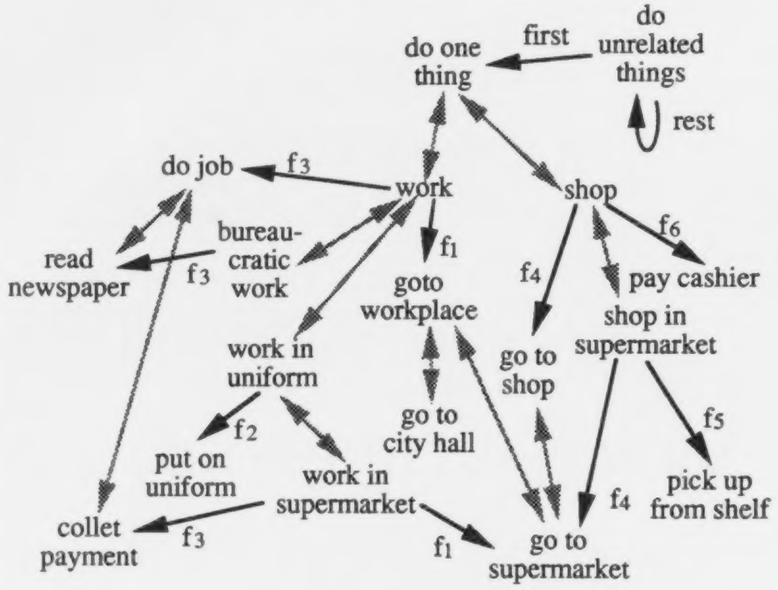

do one thing
first
do unrelated things
rest
do job
f3
work
shop
f6
read newspaper
bureau-cratic work
f3
f1
goto workplace
f4
pay cashier
shop in supermarket
work in uniform
go to shop
go to city hall
f5
f2
put on uniform
work in supermarket
pick up from shelf
f4
collet payment
f3
f1
go to supermarket

Figure 42: A working/shopping world

For example, Figure 42 shows a CF-Diagram of a fraction of working/shopping world. We briefly explain the knowledge encoded in the CF-Diagram:

- To work, one first goes to the work place and then does whatever job he/she is suppose to do. Thus the feature links from work to goto-workplace and do-job. For bureaucrats, doing their job means reading comics in newspapers.
- Some kinds of work require the workers to wear uniforms. A supermarket worker's job is one of them.
- A supermarket worker goes to the supermarket to work and the job is to collect payment.
- To shop, one first goes to a shop, get the merchandise, and then pays the cashier.
- Shopping in supermarket is a kind of shopping, where one goes to a supermarket and gets the merchandise by picking it from the shelves.

Once a CF-Diagram is created, we can use it to understand the stories about the domain. Consider the following example from [44]:

> "John went to the supermarket. He put on the uniform."

The two sentences are represented as two observations:

$$\begin{aligned}&\exists x.\ \ \texttt{go-to-supermarket}(x) \wedge \texttt{agent-name}(x) = \texttt{John}\\&\exists x.\ \ \texttt{put-on-uniform}(x) \wedge \texttt{agent-gender}(x) = \texttt{male}\end{aligned}$$

Two scenarios explaining the observations are shown in Figure 43. The scenario in (1) explains the observation by postulating that John is going to supermarket to work. The scenario (2), on the other hand, says that John went to the supermarket to shop and some man (not necessarily John) put on uniform to work. Assuming:

$$\begin{aligned}&[\texttt{goto-supermarket}(f_1(x))|\texttt{work-in-supermarket}(x)]_x = 1\\&[\texttt{put-on-uniform}(f_2(x))|\texttt{work-in-uniform}(x)]_x = 1\\&[\texttt{goto-supermarket}(f_4(x))|\texttt{shop-in-supermarket}(x)]_x = 1\\&[\texttt{do-one-thing}(first(x))|\texttt{do-unrelated-things}(x)]_x = 1\end{aligned}$$

Then the probability of scenario (1) is

$$[\texttt{work-in-supermarket}(x)]_x$$

whereas the probability of scenario (2) is

$$\begin{aligned}
&[\texttt{do-unrelated-things}(x)]_x \times \\
&[\texttt{do-unrelated-things}(rest(x))|\texttt{do-unrelated-things}(x)]_x \times \\
&[\texttt{shop-in-supermarket}(x)|\texttt{do-one-thing}(x)]_x \times \\
&[\texttt{work-in-uniform}(x)|\texttt{do-one-thing}(x)]_x
\end{aligned}$$

Therefore, the first scenario tends to be more probable than the second unless `work-in-supermarket` is an extremely unlikely event.

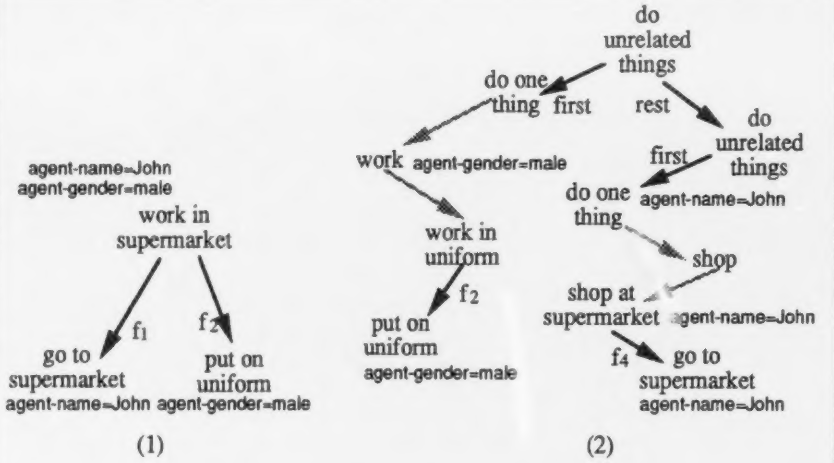

agent-name=John
agent-gender=male
work in supermarket
f1
f2
go to supermarket
agent-name=John
put on uniform
agent-gender=male
(1)
do unrelated things
do one thing
first
rest
do unrelated things
work
agent-gender=male
first
do one thing
agent-name=John
work in uniform
f2
shop
put on uniform
agent-gender=male
shop at supermarket
agent-name=John
f4
go to supermarket
agent-name=John
(2)

Figure 43: "John went to the supermarket. He put on the uniform"

The CF-Diagram in Figure 42 can also be used to understand many other stories in the same domain:

- From "Sue went to the shop, she picked up bottle of milk from the shelf and paid the cashier," we may infer that Sue bought a bottle of milk in a supermarket (Figure 44). Note that the story did not say that the shop Sue went to is a supermarket. However, since Sue picked up the milk from the shelf herself, instead of asking a shop assistant to do it for her, we can infer that she was shopping in a supermarket.

- From "Bill went to the city hall and began to read the newspaper," we may infer that Bill is a bureaucrat working in the city hall (See Figure 45). There is no direct explanation of going to city hall, the observation inherits the explanation from `going-to-work-place`.

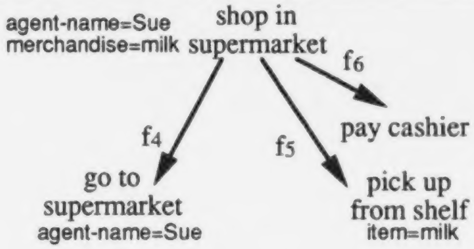

agent-name=Sue
merchandise=milk
shop in supermarket
f6
pay cashier
f4
f5
go to supermarket
agent-name=Sue
pick up from shelf
item=milk

Figure 44: Buy milk scenario

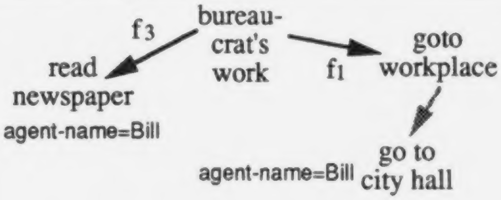

f3
bureau-crat's work
read newspaper
agent-name=Bill
f1
goto workplace
go to city hall
agent-name=Bill

Figure 45: Bureaucrat in city hall

6.2. Comparison of abduction theories

This section compares obvious abduction with the abduction theories reviewed in the last section. The major characteristics being compared include: representation of domain knowledge, problem formulation, explanation ranking, and computational complexity.

6.2.1. Knowledge representation

Figure 46 shows a classification of abduction theories according to how they represent domain knowledge.

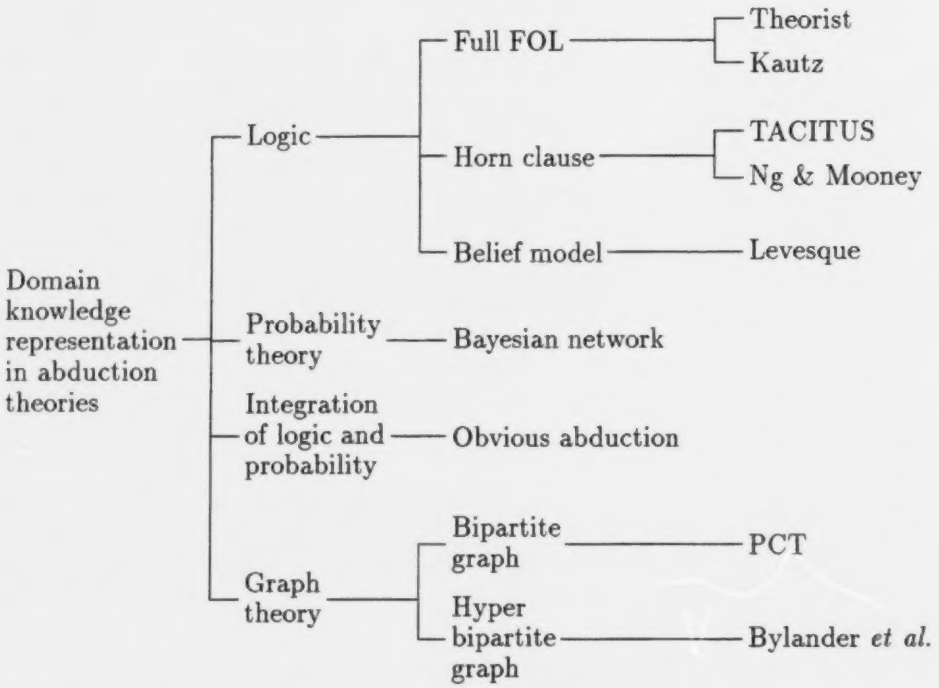

Domain knowledge representation in abduction theories
Logic
Full FOL
Theorist
Kautz
Horn clause
TACITUS
Ng & Mooney
Belief model
Levesque
Probability theory
Bayesian network
Integration of logic and probability
Obvious abduction
Graph theory
Bipartite graph
PCT
Hyper bipartite graph
Bylander *et al.*

Figure 46: Representation of domain knowledge in abduction

A good knowledge representation scheme for a chosen domain (or domains) should possess the follow two properties:

Descriptive adequacy: the ability to capture the essential knowledge in the chosen domain (or domains).

Procedural adequacy: the ability to support efficient processes of inference and search.

The procedural adequacy of the abduction theories is dealt with in Section

6.2.4. Here we are concerned with the descriptive adequacy of the abduction theories. Specificly, we discuss how well the formalisms represent constraints, taxonomic hierarchy, and multi-cause interaction in abduction.

Constraints

Constraints play an important role in abduction. Since explanations must be consistent with the knowledge base, the constraints in the knowledge base can be used to rule out impossible explanations. For example, suppose we have a constraint stating that the misfiring caused by bad-spark has regular occurrence, then if the symptom of misfiring is intermittent, the possibility of bad-spark is ruled out. We have shown how such constraints are represented in obvious abduction. Since the constraint axioms in obvious abduction are a subset of FOL, the constraints can also be expressed in the same way in logic-based approaches.

To represent such constraints in Bayesian networks, one can define a node to represent a random variable X which takes value from

$$\{\texttt{regular-misfiring}, \texttt{irregular-misfiring}, \texttt{no-misfiring}\}$$

The constraint can be represented by the conditional probability

$$P(\texttt{irregular-misfiring}|\texttt{bad-spark}) = 0$$

However, this representation does not allow the observation `misfiring` without specifying whether its occurrences are regular or not.

In the set covering model, any explanation is consistent with knowledge base because the theory does not provide mechanisms for specifying constraints. There is no mechanism to prevent an explanation including two mutually exclusive disorders.

Taxonomic hierarchy

The main goal of taxonomic hierarchy is to allow inheritance reasoning. Inheritance has been described as "the primary, most powerful representation primitive available in knowledge representation [16]."

A set covering model is unable to represent taxonomic hierarchy. In Theorist and other logic-based approaches, taxonomic hierarchy is represented by material implications. We believe that taxonomic relationship is more than material implication. For example, if "A is-a B", then an explanation of A may also serve as an explanation of B. This is not true of logical implication in general. Consider the following two sentences:

$$\forall x.\texttt{drive-blue-Mercedes}(x) \rightarrow \texttt{drive-Mercedes}(x)$$
$$\forall x.\texttt{own-Mercedes}(x) \rightarrow \texttt{drive-Mercedes}(x)$$

Since `steal-Mercedes(John)` explains `drive-Mercedes(John)`, this explanation can be inherited by `drive-blue-Mercedes(John)`, i.e., `steal-Mercedes(John)` also explains `drive-blue-Mercedes(John)`. However, this explanation cannot be inherited by `own-Mercedes(John)`. In the first case, `drive-blue-Mercedes` is a `drive-Mercedes`, whereas in the second case `own-Mercedes` simply implies `drive-Mercedes`.

Kautz's plan hierarchy contains taxonomic hierarchies. However, the treatment of specificity is informal. Specificity of an explanation is controlled by a `consider-spec` flag in the plan recognition algorithm. There is no declarative specification of how this mechanism should work.

Pearl [47] showed that Bayesian networks can be used to represent taxonomic hierarchy. However, in [47], a Bayesian network can contain either "isa" link or causal links, but not a mixture of them.

This is because the links in a Bayesian network have homogeneous semantics. However, an "isa" link $a \overset{\text{isa}}{\rightarrow} b$ cannot be treated in the same way as a link for which $P(b|a) = 1$. Comparing (1) and (2) in Figure 47, for example, the influence

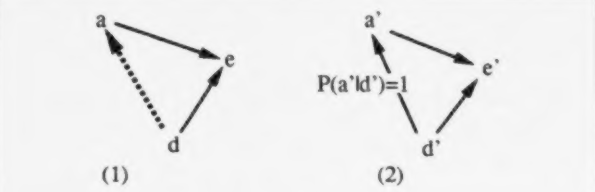


Figure 47: Isa link must be treated differently

of d on e via a should be preempted by the direct link from d to e, whereas in (2), the influence of d' on e' via the two paths should be combined.

Multi-cause interaction

Abductive explanation usually describes a causal process. One of the complicating issues in causal processes is the interaction between multiple causes. That is, the effects of simultaneous causes may not simply be the union of their individual effects. Consider the following example:

> Diarrhea and vomiting both cause substantial loss of body potassium. Thus, together, their effect on hypokalemia is compounded. On the other hand, diarrhea results in loss of alkalis, vomiting results in loss of body acids. Therefore, taken together they tend to offset each other's effect on serum acidity. [46, p.367]

In Bayesian networks, the multi-cause interaction is modeled by the conditional probability of an effect e given its direct causes $c_1, c_2, \ldots, c_n$:

$$P(e|c_1, c_2, \ldots, c_n)$$

If the interaction can be modeled by a **noisy-or-gate**, then the following equation holds:

$$P(\bar{e}|c_1, c_2, \ldots, c_n) = P(\bar{e}|c_1) \times P(\bar{e}|c_2) \times \ldots \times P(\bar{e}|c_n)$$

Intuitively, **noisy-or-gate** means that exceptions to the causal relationships are independent. In a noisy or gate, any member of the direct causes $c_1, c_2, \ldots, c_n$ is likely to cause the effect e and the likelihood does not diminish when several of these conditions prevail simultaneously.

PCT assumes that the multi-cause interaction is **noisy-or-gate**. Although it is one of the most common forms of multi-cause interaction, the **noisy-or-gate** fails to be an accurate model in many cases. The approach proposed by Bylander *et al.* allows representation of cancellation of effects as well as additional effects that result from simultaneous presence of interacting causes.

In obvious abduction, multi-cause interaction is generally modeled by noisy or gate. When the multi-cause interaction is not noisy or gate, one has to explicitly create a category which is subsumed by the interacting causes to represent the interaction. For example, the interaction between diarrhea and vomiting is not noisy or gate. We create a category `diarrhea&vomiting` to represent the interaction (Figure 48). The category `diarrhea&vomiting` is a subclass of `diarrhea` and `vomiting`, therefore, it can inherit properties from the two superclasses. On the other hand, the path preemption mechanism also allows `diarrhea&vomiting` to override the properties of its superclasses. In Figure 48, the causal link between `diarrhea&vomiting` and `high-serum-acidity` is associated with a low probability representing the cancellation effect of the interaction between `diarrhea` and `vomiting`.

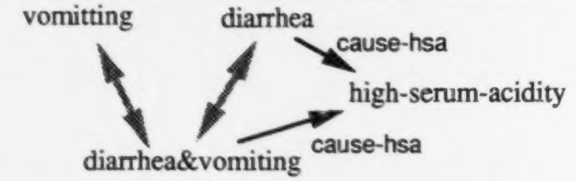


Figure 48: Representing multi-cause interaction in obvious abduction

The drawback of this approach is that a category has to be explicitly created for each set of interacting causes. The features of the category is the union of the interacting causes. As a result, such categories may have a large number of features which are potentially expensive for the inference algorithm, because its complexity is exponential in the maximum number features of a category.

Summary

Obvious abduction is more expressive than PCT in all the aspects we have discussed. Obvious abduction is weaker than logic-based and probability-based approaches w.r.t. the representation of multi-cause interaction. It is stronger than Bayesian network but weaker than logic in the representation of constraints; it is stronger than both logic and probability based approaches in the representation of taxonomic hierarchy. The advantage of obvious abduction is that where it is weaker than other approaches, it still seems to be adequate to represent the knowledge in the domains we are concerned with. Obvious abduction is the first approach that integrates causal, probabilistic and taxonomic information in abduction.

6.2.2. Problem formulation

The definitions of explanation in abduction theories can be categorized as follows:

Minimal hypotheses as explanation: An explanation of the observations is a set of hypotheses that satisfies certain minimality criteria and logically implies the observations.

Coherent hypotheses as explanation: An explanation is coherent set of hypothesis that logically implies the observations.

Model as explanation: An explanation of the observations is a model in which the observations are true.

Covering as explanation: An explanation of the observations is a cover of the observations, where the covering relation is defined in terms of causal capability.

Figure 49 shows a hierarchical classification of domain-independent abduction theories according to their definitions of explanation.

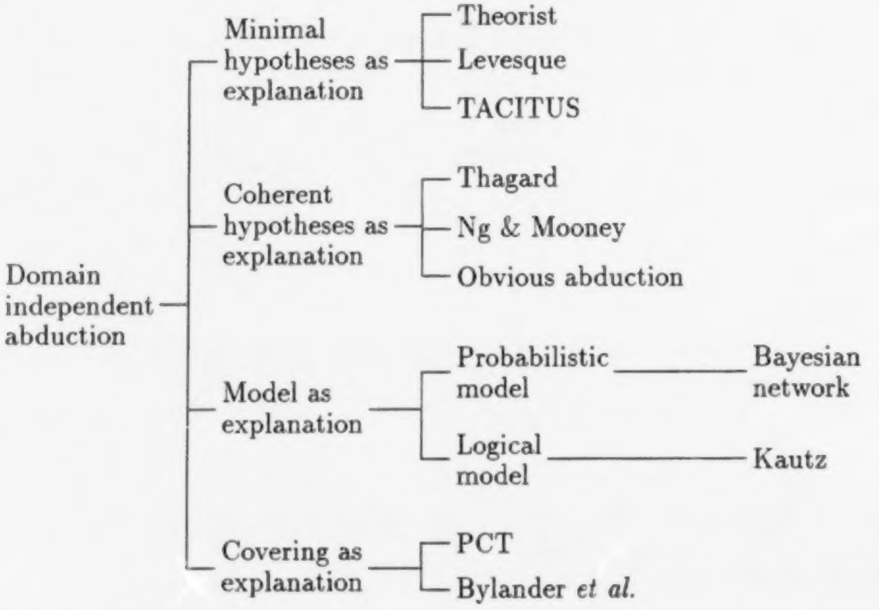

Domain independent abduction
Minimal hypotheses as explanation
Theorist
Levesque
TACITUS
Coherent hypotheses as explanation
Thagard
Ng & Mooney
Obvious abduction
Model as explanation
Probabilistic model
Bayesian network
Logical model
Kautz
Covering as explanation
PCT
Bylander *et al.*

Figure 49: Definitions of explanation

The disadvantage of defining an explanation as a minimal set of hypotheses or a cover of the observations is that coherence of explanation is not inherent in the definition. The hypotheses can be totally unrelated to each other as long as they logically imply the observations. Secondly, uncovering the linkage from the hypotheses to the observations is part of the abduction. For example, in parsing, the goal is not only to show that a sentence category dominates a sequence of words, but also to uncover the detailed structure of the dominance relation—the parse tree.

The disadvantage of models as explanations is that models often have to be overly detailed. For example, if `take-vacation(John)` is part of a model and there are 100 possible destinations for vacations. The model must commit to one of the 100 places. A complete variable assignment in a Bayesian network suffers from the same problem. An additional unsatisfactory aspect of complete variable assignment is that it assigns values to all the variables in the Bayesian network, even if many of them may be only remotely related to the current situation.

6.2.3. Explanation ranking

Explanation ranking is the most important issue in abduction. The credibility of abductive conclusion depend on this. Figure 50 shows a classification of explanation ranking in abduction theories.

Abduction is unsound inference. Unlike deduction, the conclusion of abductive inference does not necessarily follow from the observations it explains. It has been shown by Cox [13] that if the reasoning agent is not acting against it own interest, the choice it makes should be the most probable one. Therefore, should the probabilities of the alternative explanations be available, a rational agent should choose the most probable explanation. Minimality of the hypotheses set is a likely characteristic of the most probable explanation. However, there are cases where the best explanation is not minimal. For example: Figure 51 depicts the causal

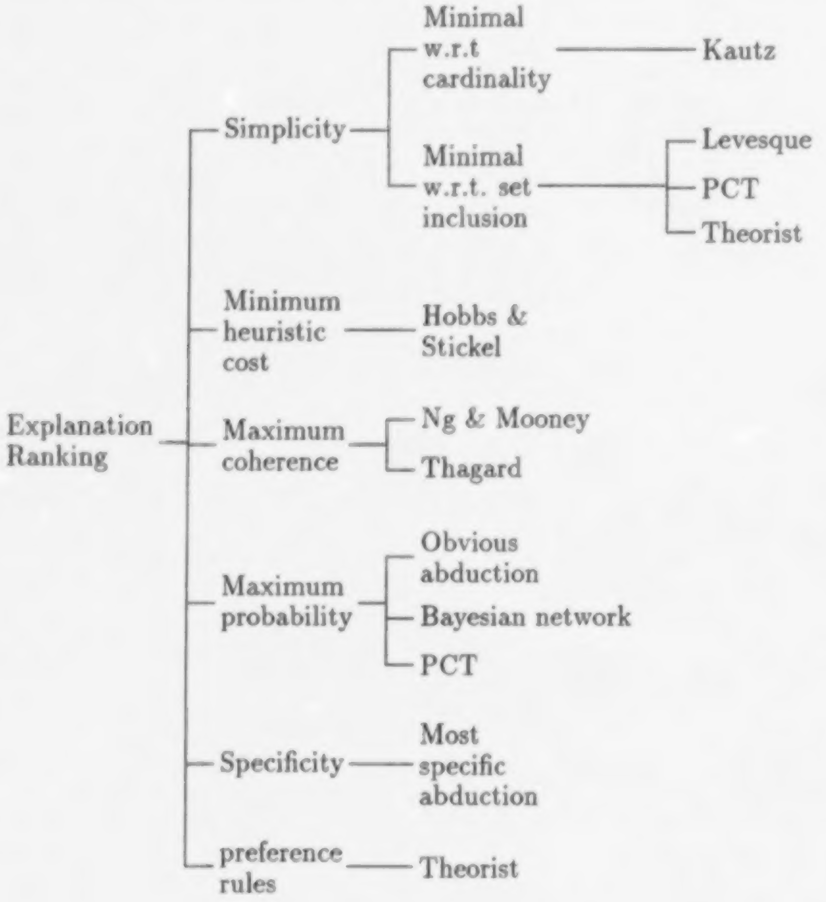

Explanation Ranking
Simplicity
Minimal w.r.t cardinality
Kautz
Minimal w.r.t. set inclusion
Levesque
PCT
Theorist
Minimum heuristic cost
Hobbs & Stickel
Maximum coherence
Ng & Mooney
Thagard
Maximum probability
Obvious abduction
Bayesian network
PCT
Specificity
Most specific abduction
preference rules
Theorist

Figure 50: Explanation ranking in abduction

relationship between disorders $\{d_1, d_2\}$ and symptoms $\{s_1, s_2, s_3\}$. Suppose s_2 and s_3 are typical symptoms of d_2, and d_1 causes $\{s_2, s_3\}$ only in rare cases. When s_1, s_2 and s_3 are observed, the explanation that d_1 caused s_1 and d_2 caused $\{s_2, s_3\}$ is intuitively better than the explanation that d_1 caused $\{s_1, s_2, s_3\}$, even though the latter consists a smaller set of hypotheses.

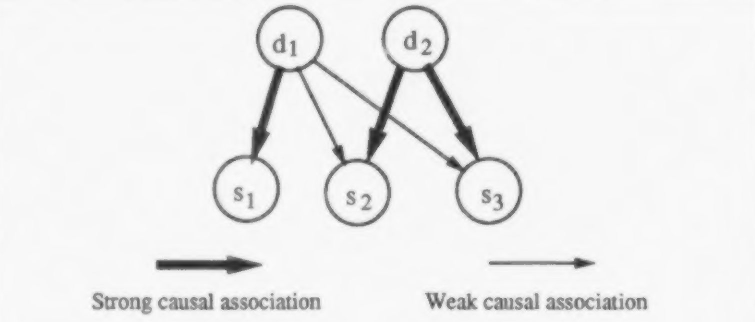


Figure 51: Most probable explanation may not be minimal

The disadvantage of minimal hypothesis approaches has been most eloquently argued by Charniak and Goldman [7].

> [Kautz's approach] cannot decide that a particular plan, no matter how likely, explains a set of actions, as long as there is another plan, no matter how *unlikely*, which could also explain the observed action. Consider
>
> > Jack packed a bag. He went to the airport.
>
> any normal reader would assume that Jack is taking a plane-trip. but Kautz's plan-recognition system would not be able to decide between plane-trip, and air-terrorist-bombing, since the latter also has the terrorist packing a bag (with a bomb) and going to the airport (to get the bag on the plane).

A commonly cited objection to the use of probability is that the numerical probabilities are seldom available. We believe that the probability-based approaches do not require close estimation of probabilities to be functional. This is because if close estimates are required to distinguish between hypotheses, then they are not much better than on another. Therefore, these cases should be regarded as true ambiguities which can only be resolved by further evidence. On the other hand, if one hypothesis is substantially more probable than another, then even rough estimates will result in enough difference in the likelihood of the two hypotheses to make preferences.

6.2.4. Computational complexity

Generally speaking, the more expressive a theories is, the more difficult the reasoning in that theory. As we have discussed earlier, PCT is one of the least expressive theories. Yet, abductive reasoning in PCT is NP-complete. The complexities of the abductive reasoning theories are summarized in Table 6.2.

Table 6.2: Complexities of abduction theories

Theory	Complexity
Set covering model	NP-hard
Theorist	Undecidable
Kautz's theory	NP-hard
Bayesian Network (singly connected)	Polynomial
Bayesian Network (multiply connected)	NP-hard

The only theory that is tractable is abductive reasoning in singly connected Bayesian Networks. However, restricting Bayesian network to be singly connected severely limit the representational capability. Many commonly used knowledge cannot be represented in such networks. Compared with other abduction theories, obvious abduction take advantage of the characteristics in the input. We have shown in Chapter 4 that the complexity of obvious abduction is:

$$O(M^2 3^{k+d} |\mathrm{CF_{KB}}|)$$

where $|\mathrm{CF_{KB}}|$: the total number of links in the CF-Diagram;

k: the number of observations to be explained;

d: the maximum number of features of a category;

M: the number of possible attribute value assignments.

In the applications we are considering, the number of observations is usually small. The features represent direct structural relationships between the objects in the domain. Although an object may be related to a large number of objects, those that are directly related to it tend to be small in number. Therefore, obvious abduction can be performed efficiently.

We also note that the complexity of a particular application of obvious abduction can be much better than the general worst case. For example, the complexity of CFG parsing is polynomial in the number of observations (the number of words in the input sentence).

6.3. Spreading activation in semantic networks

The obvious abduction explains a set observations by finding a connection between them in the CF-Diagram. This is similar to spreading activation in semantic networks.

Spreading activation is one of the earliest forms of reasoning in semantic networks. The questions answered by spreading activation would be very difficult to answer for a knowledge base that was not in semantic-network form.

Viewing a semantic network as a graphical representation of logical constructs is not original in this work. The original contribution of this research is that the inference made from the network has also been formally specified.

> "It is unfortunately much easier to develop an algorithm that appears to reason over structures of a certain kind that to *justify* its reasoning by explaining what the structures are saying about the world [37]."

Our improvements over the early spreading activation are as follows:

1. The formalization of the structures inferred by the graph searching algorithm. For example, the intersection search in semantic networks has only been informally described as a procedure to find associations between two concepts. Questions like

- "What exactly do these associations mean?"
- "Which association is valid or invalid?"
- "How and where should they be used?"

are not formally addressed. In our representation scheme, the observations and scenarios are also logical formulas with logical semantics.

2. The use of formal probability in place of heuristic weights in graph searching. The numbers associated with the nodes and links in CF-Diagrams are statistically meaningful. In contrast, there is no semantic foundation for the weights often associated with nodes and links in semantic networks.

3. We also generalize the path finding to walk-tree finding. Spreading activation is usually a search for the connection between **two** concepts, whereas the scenarios found by our message passing algorithm is a connection amoung possibly more than two concepts.

4. Although spreading activation and inheritance reasoning is very common with semantic networks, no serious attempt has been made to combine spreading activation and inheritance reasoning so that the connections found by the spreading activation are maximally specific with respect to the concepts being connected.

In summary, obvious abduction may be viewed as a formalization and generalization of spreading activation.

Chapter 7

Conclusions

7.1. Summary of Contributions

We have presented a unified theory of abduction that is applicable across several domains. The knowledge representation language in obvious abduction provides mechanisms for representing taxonomic hierarchies, constraints, and probabilistic information. The formulation of the abduction problem is more general in that any type of events can be observations. For example, in diagnosis, faults or intermediate states can also be observations; in plan recognition, observations can not only be basic actions, but also high-level plans. In contrast, in many other approaches, observations can only be symptoms or basic actions.

Specificity is a confusing issue in abduction. Previous approaches rely on apparently contradictory intuitions. We clarify this issue by identifying the appropriate places for using the intuitions and present a formal treatment of specificity in abduction.

One of the most important issues in abduction is the ranking of explanations. Unlike many other approaches, which use heuristic measurements of simplicity or coherence to rank explanations, explanation ranking in obvious abduction is based on formal probability theory. The definition of best explanation also incorporates considerations for consistency, coherence, relevance and specificity.

We presented an efficient message passing algorithm for abductive inference. The adaptation of the algorithm to different domains amounts to different instantiations of the generic message structure and propagation constraints. The object-oriented nature of the algorithm not only lends itself to distributed parallel processing, but also facilitates extension and adaptation of the algorithm in different application domains.

Complexity is a major obstacle in practical applications of abduction. Even theories with modest expressive power have been proved to be computationally intractable. The inference algorithm for obvious abduction takes advantages of the admissible input, so that it is polynomial with respect to the size of the knowledge base and exponential in the number of observations to be explained.

Applications of obvious abduction in diagnosis, plan recognition and parsing also shed some new light on problem-solving in these areas:

Diagnosis: Causal, taxonomic and probabilistic knowledge have been recognized to be the most important kinds of knowledge in diagnostic reasoning. Efforts have been made to integrate each pair of the three. To the best of author's knowledge, this is the first formal approach where all three kinds of knowledge can be represented and reasoned with.

The representation of causation in obvious abduction is more general than in many other approaches in that causations can exist between any two events, including faults, symptoms, and intermediate states; cyclic causal tendencies can also be represented in obvious abduction.

Plan recognition: Probabilistic knowledge allows rational choice to be made in case of ambiguities. The plan representation scheme is restrictive enough for the inference to be tractable, and meanwhile is expressive enough to handle partially ordered, interleaved and recursive plan steps.

The ability to recognize recursive plans also allows us to investigate a novel application of plan recognition: algorithm recognition.

Parsing: We showed that CFG parsing can be cast as an abductive reasoning problem. The complexity of our algorithm is $O(|G|n^3)$, where n is the length of the input sentence and $|G|$ is the size of the grammar. In contrast, most other general CFG parsers, such as Earley's algorithm and chart parsers, have the complexity $O(|G|^2n^3)$. This is significant improvement in natual language parsing where the size of grammar tends to be large.

7.2. Future Work

The future research on obvious abduction consists of the continued development of the general framework and the further investigation of its applications.

7.2.1. The general framework

The credibility of an explanation depends not only on its own merits, but also on the qualities of the alternative explanations. A future extension of the general framework of obvious abduction includes a more sophisticated representation of alternative explanations.

A set of observations is truly ambiguous if there are multiple explanations with approximately the same probability. Most previous approaches simply list all the alternative explanations. A representation scheme for the alternative explanations should highlight the source of ambiguities, and allow sharing of common components in the alternative explanations. The motivation for such a representation is that the highlighted source of ambiguities will provide clues as to what crucial questions will disambiguate among the alternatives. The shared common components will make it easier for the abductive reasoner to conclude whatever is true in all the alternative explanations. Such a scheme already exists in parsing, known as the "shared packed parse forest" [77], which will provide the basis of explanation representation in a more general context.

7.2.2. Diagnosis

In the monitoring and diagnosis of a large physical system, such as a ship or a computer network, there may be a large number of observations. The complexity of obvious abduction increases exponentially with the number of observations. One possible way to handle a large number of observations is to prioritize the observations and deal with the most important observations first. Another possibility is to divide the domain hierarchically into sub-domains. This way, a large number of observations will be distributed in a large number sub-domains with only a small number of observations in each sub-domain. We can apply the algorithm for obvious abduction in the sub-domains and treat the conclusions in each sub-domain as observations at a higher level.

Diagnosis is an iterative process. In the case of multiple equally-good explanations, a diagnostic system has to make new observations in order to reduce the possible explanations. An important question is how to guide the abductive reasoner to gather further evidence. This is a similar problem to identifying crucial literals in Theorist [65].

7.2.3. Plan recognition

Currently, the plan library for algorithm recognition consists only of variations of quick sort and insertion sort. Further support of the claim that obvious abduction is a viable solution for the algorithm recognition problem needs expansion of the plan library, e.g, to include other sorting algorithms, and operations on other data structures.

7.2.4. Natural language parsing

We noted that the CFG parsing algorithm can be extended to handle context-sensitive features of natural language, such as agreement. Further proof of capability of obvious abduction in natural language parsing needs a full-fledged implementation of a natural language grammar theory. R-GPSG[1] is a good candidate for the following reasons:

- R-GPSG is the result of a careful study of GPSG to reduce its complexity while sacrificing as little expressive power as possible.
- R-GPSG is similar to obvious abduction in that the attribute values are atomic, whereas the attribute values in GPSG and other information based linguistic theories are arbitrary acyclic graphs.
- To the best of author's knowledge, R-GPSG has not been implemented up to now.

---

[1] R-GPSG [3] is a restricted version of GPSG (Generalized Phrase Structure Grammar) [17]

Bibliography

[1] J. F. Allen and C. R. Perrault. Analyzing intention in dialogues. *Artificial Intelligence*, 15(3):143–178, 1980.

[2] Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge.* PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 1988.

[3] G. Edward Barton, Jr., Robert C. Berwick, and Eric Sven Ristad. *Computational Complexity and Natural Language.* Computational Models of Cognition and Perception. The MIT Press, Cambridge, Massachusetts, 1987.

[4] Alan Bundy. Incidence calculus: a mechanism for probabilistic reasoning. *Journal of Automated Reasoning*, 1:263–283, 1985.

[5] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. Some results concerning the computational complexity of abduction. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *Proceedings of First International Conference on Principles of Knowledge Representation and Reasoning*, pages 44–54, 1989.

[6] Sandra Carberry. Modeling the user's plan and goals. *Computational Linguistics*, 14(3):23–37, 1988.

[7] Eugene Charniak and Robert Goldman. A probabilistic model of plan recognition. In *Proceedings of the Ninth national Conference on Artificial Intelligence*, pages 160–165. AAAI, 1991.

[8] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley Publishing Company, 1985.

[9] A. M. Collins and E. F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82:407–428, 1975.

[10] Luca Console, Daniele Theseider Dupre', and Pietro Torasso. A theory of diagnosis for incomplete causal models. In *Proceedings of IJCAI89*, pages 1311–1317. Morgan Kaufmann Publishers, Inc., 1989.

[11] G. Cooper. *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge.* PhD thesis, Department of Computer Science, Stanford University, November 1984.

[12] P. T. Cox and T. Pietrzykowski. General diagnosis by abductive inference. In *Proceedings of the 1987 Symposium on Logic Programming*, pages 183–189, 1987.

[13] R. R. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1), 1946.

[14] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[15] Tim Finin and Gary Morris. Abductive reasoning in multiple fault diagnosis. Technical Report LINC Lab 123, Department of Computer and Information Science, University of Pennsylvania, 1988.

[16] Mark S. Fox. On inheritance in knowledge representation. In *Proceedings of IJCAI-79*, pages 282–284, 1979.

[17] Gerald Gazdar, Ewan Klein, Geoffery Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Basil Blackwell Publisher Ltd, Oxford, UK, 1985.

[18] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1987.

[19] R. Goebel and S.D. Goodwin. Applying theory formation to the planning problem. In *Proceedings of the AAAI Workshop on The Frame Problem in Artificial Intelligence*, pages 207–232, Lawrence, Kansas, April 12-15 1987.

[20] Randy Goebel. A logic data model for the machine representation of knowledge. Technical Report CS-86-07, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1986.

[21] Robert P. Goldman and Eugene Charniak. Dynamic construction of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97, 1990.

[22] B. Goodman and D. Litman. Plan recognition for intelligent interfaces. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications*, pages 297–303, 1990.

[23] Scott Goodwin. Semantic tools for the analysis and synthesis of a vocabulary for hypothetical reasoning. Ph.d thesis proposal, Department of Computing Scinece, University of Alberta, July 1989.

[24] Scott Goodwin and Randy Goebel. Nonmonotonic reasoning in temporal domains: The knowledge independence problem. In *Proceedings of the Second International Workshop on Nonmonotonic Reasoning*, pages 187–201, January 1989.

[25] Benjamin N. Grosof. Non-monotonicity in probabilistic reasoning. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 237–249. North-Holland, 1988.

[26] William Havens and Alan Mackworth. Representing and using knowledge of the visual world. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier*, chapter 16, pages 429–450. Springer-Verlag, 1987.

[27] Jerry R. Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. Interpretation as abduction. Technical Report 499, SRI International, Menlo Park, CA, December 1990.

[28] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of 26th Annual Meeting of the Association for Computational Linguistics*, pages 95–103, Buffalo, 1988. ACL.

[29] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and Their Relation to Automata.* Addison-Wesley Publishing Company, 1969.

[30] Michael C. Horsch and David Poole. A dynamic approach to probabilistic inference using Bayesian networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 155–161, 1990.

[31] W. Lewis Johnson. *Intention-Based Diagnosis of Novice Programming Errors.* Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[32] John R. Josephson. On the "logical form" of abduction. In *Working Notes of AAAI Symposium on Automated Abduction*, pages 140–144, 1990.

[33] S. J. Kaplan. Designing a portable natural language database query system. *ACM Trans. on Database Systems*, 9 (1):1–19, March 1984.

[34] Henry A. Kautz. A formal theory of plan recognition. Technical Report 215, Department of Computer Science, University of Rochester, May 1987.

[35] Hector J. Levesque. A knowledge-level account of abduction. In *Proceedings of IJCAI89*, pages 1061–1067, Detroit Mi., 1989.

[36] Hector J. Levesque. Logic and the complexity of reasoning. Technical Reports on Knowledge Representation and Reasoning KRR-TR-89-2, Department of Computer Science University of Toronto, 1989.

[37] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3(2):78–93, May 1987.

[38] Dekang Lin and Randy Goebel. Computing circumscription of ground theories with theorist. Technical Report 89–26, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, October 1989.

[39] Dekang Lin and Randy Goebel. A minimal connection model of abductive diagnostic reasoning. In *Proceedings of 1990 IEEE Conference on Artificial Intelligence Applications*, pages 16–22, Santa Barbara, California, March 1990.

[40] D. Litman and J. Allen. Discourse processing and commonsense plans. In *Intensions in Communication*. MIT Press, 1991.

[41] Diane J. Litman. Plan recognition and discourse analysis: An integrated approach for understanding dialogues. Technical Report TR–170, Department of Computer Science, University of Rochester, 1985.

[42] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13, 1980.

[43] R. Miller, H. Pople, and J. Myers. INTERNIST-I, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[44] Hwee Tou Ng and Raymond J. Mooney. On the role of coherence in abductive explanation. In *Proceedings of AAAI-90*, pages 337–342, 1990.

[45] Nils J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–88, Feb. 1986.

[46] Ramesh S. Patil. Artificial intellignece techniques for diagnostic reasoning in medicine. In Howard E. Shrobe, editor, *Exploring Artificial Intellignece*, chapter 9, pages 347–379. AAAI Press, 1988.

[47] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[48] Judea Pearl and T. S. Verma. A theory of inferred causation. In *Proceedings of Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 441–452, 1991.

[49] C. S. Peirce. *Essays in the Philosophy of Science*.

[50] Yun Peng and James A. Reggia. A probabilistic causal model for diagnostic problem solving — Part I: integrating symbolic causal inference with numeric probabilistic inference. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-17(2):146–162, 1987.

[51] Yun Peng and James A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving*. Symbolic Computation. Springer-Verlag, 1990.

[52] Martha E. Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, University of Pensylvania, 1986.

[53] Martha E. Pollack. Some requirements for a model of the plan inference process in conversation. In R. G. Reilly, editor, *Communications Failure in Dialogue and Discourse*, pages 245–256. Elsevier Science Publishers, 1987.

[54] D. L. Poole. On the comparison of theories: Preferring the most specific explanation. In *Proceedings of IJCAI85*, pages 144–147, Los Angeles, California, August 16-18 1985. UCLA.

[55] David Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36, 1988.

[56] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352. Springer-Verlag, 1987.

[57] Harry. E. Pople, Jr. On the mechanization of abductive logic. In *Proceedings of IJCAI-73*, pages 147–152, Stanford, CA, 1973.

[58] Harry E. Pople, Jr. Evolution of an expert system: from Internist to Caduceus. In I. De Lotto and M. Stefanelli, editors, *Artificial Intelligence in Medicine*, pages 179–208. Elsevier Science Publishers, 1985.

[59] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*. The MIT Press, Cambridge, MA, 1968.

[60] Reiter Raymond. A logic for default reasoning. *Artificial Intelligence*, 13(1&2):81–132, 1980.

[61] J. Reggia, D. S. Nau, and P. Y. Wang. Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, 19:437–460, 1983.

[62] Raymond Reiter. Nonmonotonic reasoning. In *Annual Review of Computer Science*, pages 147–186. Annual Reviews Inc., 1987.

[63] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems. In *Proceedings of IJCAI87*, pages 183–187, 1987.

[64] P. A. Sachs, A. M. Paterson, and M. H. M. Turner. Escort—an expert system for complex operations in real time. *Expert Systems*, 3(1):22–29, January 1986.

[65] A. Sattar and R. Goebel. Using crucial literals to select better theories. *Computational Intelligence*, 7(1):11–22, February 1991.

[66] Abdul Sattar and Randy Goebel. Using crucial literals to select better theories. Technical Report 89–27, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, October 1989.

[67] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson. The plan recognition problem: An intersection of psychology and Artificial Intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[68] Soloman E. Shimony. Explanation, irrelevance and statistical independence. In *Proceedings of AAAI-91*, pages 482–487, Anahiem, California, July 1991.

[69] Solomon E. Shimony and Eugene Charniak. A new algorithm for finding MAP assignments to belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 98–103, 1990.

[70] Yoav Shoham. *Reasoning about Change*. The MIT Press, Cambridge, Massachusetts, 1988.

[71] Brian Skyrms and William L. Harper, editors. *Causation, Change, and Credence*. Kluwer Academic Publishers, 1988.

[72] Mark E. Stickel. A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Technical Report 451, SRI International, September 1988.

[73] Mark E. Stickel. A method for abductive reasoning natural-language interpretation. In *Working Notes of AAAI Symposium on Automated Abduction*, pages 140–144, 1990.

[74] Patrick Suppes. *Probabilistic Metaphysics*. Basil Blackwell Inc., Oxford, UK, 1984.

[75] Paul Thagard. *Computational Philosophy of Science*. The MIT Press, Cambridge, Massachusetts, 1988.

[76] Paul Thagard. Explanatory coherence. *Behavioral and Brain Sciences*, 12:435–502, 1989.

[77] Masaru Tomita. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Norwell, Massachusetts, 1986.

[78] David Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann, 1986.

[79] Marc Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of AAAI-90*, pages 190–197, Boston, MA, July 1990. The MIT Press.

[80] Sholom M. Weiss, Casimir A. Kulikowski, Saul Amarel, and Aran Safir. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

[81] Lina Mary Wills. Automated program recognition: A feasibility demonstration. *Artificial Intelligence*, 45(1–2):113–171, 1990.

[82] Dekai Wu. A probabilistic approach to marker propagation. In *Proceedings of IJCAI-89*, pages 574–580, 1989.

Appendix A. An S^3 specification of sorting algorithms

```
(defcategory cmp
  (attributes :export (result left right)))
(defcategory exch
  (attributes :export (left right)))
(defcategory sort-list
  (attributes :export (low high))
  (local-constraints
    (ge high low)))
(defcategory quicksort
  (isa sort-list)
  (attributes :local (mid))
  (defeature p partition  ;; partition step
    (percolate mid mid)
    (percolate high high)
    (percolate low low))
  (defeature sortLow sortList
    (percolate low low)
    (percolate mid (+1 high)))
  (defeature sortHigh sortList
    (percolate mid (-1 low))
    (percolate high high)))
(defcategory partition
  (attribute
    :export (mid low high))
  (defeature sp selectPivot  ;; select pivot step
    (percolate low low)
    (percolate high high)))
(defcategory partition1
  (isa partition)
  (defeature p partition10   ;; do partition step
    (percolate mid lastSmall)
    (percolate low pivot)
    (percolate low (-1 low))
    (percolate high high))
  (defeature e exch          ;; swap in pivot
    (percolate low left)
    (percolate mid right)))
(defcategory partition10
  (attribute
    :export (lastSmall pivot low high)))
(defcategory partition11
  (isa partition10)
  (attributes :local (cmpResult sub-low sub-high))
  (defeature c cmp ;; compare pivot and low
    (percolate pivot left)
    (percolate low right)
    (percolate cmpResult result)
  (defeature e exch     ;; swap low and high
    (percolate low left)
    (percolate high right))
  (defeature r partition11
    (percolate pivot pivot)
    (percolate sub-low low)
    (percolate sub-high high)
    (percolate lastSmall lastSmall))
  (local-constraints
    (-> ((neq low high) (eq cmpResult LT))
        (eq sub-low low))
    (-> ((neq low high) (eq cmpResult LT))
        (eq sub-high 1 (-1 high)))
    (-> ((neq low high) (eq cmpResult GE))
        (eq sub-high high))
    (-> ((neq low high) (eq cmpResult GE))
        (eq sub-low (+1 low)))))
(defcategory base11
  (isa partition11)
  (local-constraints
    ((eq high low))
    (-> ((eq cmpResult GE))
        (eq lastSmall low))
    (-> ((eq cmpResult LT))
        (eq lastSmall (-1 low))))
(defcategory partition12
  (isa partition10)
  (attributes
    :local (sub-low sub-high))
  (defeature il increaseLow ;; increaseLow step
    (percolate pivot pivot)
    (percolate low low)
    (percolate high high)
    (percolate sub-low sub-low))
  (defeature dh increaseHigh ;; decreaseHigh step
    (percolate pivot pivot)
    (percolate low low)
    (percolate high high)
    (percolate sub-high sub-high))
  (defeature e exch   ;; swap new-low and new-high
      (percolate sub-low left)
      (percolate sub-high right))
    (defeature r partition12
      (percolate sub-low (-1 low))
      (percolate sub-high (+1 high)))
    (local-constraints
     (-> ((gt low high))
         (eq lastSmall sub-high)))
  (defcategory increase-low
    (attributes
      :export (pivot low high new-low)
      :local (cmpResult))
    (defeature c cmp ;; compare pivot and low
      (percolate pivot left)
      (percolate low right)
      (percolate cmpResult result))
    (defeature i increase-low
      (percolate sub-low sub-low)
      (percolate low (-1 low))
      (percolate high high)
      (percolate pivot pivot))
    (local-constraint
      (-> ((gt low high))
          (eq sub-low low))
      (-> ((eq cmpResult LT))
          (eq sub-low low))))
  (defcategory decreaseHigh
    (attributes
      :export (pivot low high new-high)
      :local (cmpResult))
    (defeature c cmp ;;
        (percolate pivot left)
      (percolate high right)
      (percolate cmpResult result))
    (defeature i decreaseHigh
      (percolate sub-high sub-high)
      (percolate low low)
      (percolate high (+1 high))
      (percolate pivot pivot))
    (local-constraint
      (-> ((gt low high))
          (eq sub-high high))
      (-> ((eq cmpResult GE))
          (eq sub-high high))))
  (defcategory partition2
    (isa partition)
    (attributes
      :local (cmpResult exchLeft exchRight sub-low sub-high))
    (defeature c  cmp ;; compare the lower two elements
      (percolate cmpResult result)
      (percolate low left)
      (percolate low (-1 right)))
    (defeature e exch ;; exch step
      (percolate exchLeft left)
      (percolate exchRight right))
    (defeature r partition2
      (percolate sub-low low)
      (percolate sub-high high))
    (local-constraints
      (-> ((eq low high)) (eq mid low))
      (-> ((neq low high) (eq cmpResult LT))
          (eq exchLeft (+1 low)))
      (-> ((neq low high) (eq cmpResult LT))
          (eq exchRight high))
      (-> ((neq low high) (eq cmpResult LT))
          (eq sub-low low))
      (-> ((neq low high) (eq cmpResult LT))
          (eq sub-high (-1 high)))
      (-> ((neq low high) (eq cmpResult GE))
          (eq exchLeft low))
      (-> ((neq low high) (eq cmpResult GE))
          (eq exchRight (+1 low)))
      (-> ((neq low high) (eq cmpResult GE))
          (eq sub-low (+1 low)))
      (-> ((neq low high) (eq cmpResult GE))
          (eq sub-high high))))
(defcategory insertionSort
  (isa sortList)
  (defeature sortLow insertionSort
    (percolate low low)
    (percolate high (+1 high)))
  (defeature l insertLast ;; insert last step
    (percolate low low)
    (percolate high high))
  (local-constraint
    (lt low high))
(defcategory insertLast
  (attributes
    :export (low high)
    :local (cmpResult))
  (defeature c cmp
    (percolate high right)
    (percolate high (+1 left))
    (percolate cmpResult result))
  (defeature e exch
    (percolate high right)
    (percolate high (+1 left)))
  (defeature i insertLast
    (percolate low low)
    (percolate high (+1 high))))
```

Appendix B. Experimental results on CFG parsing

The implementation of the CFG parsing algorithm has been tested on a SPARC-station/slc. The test grammar is Grammar III in Tomita's book [77, pp.172–6], which contains about 220 CFG rules. Since our parser is able to handle ID-rules, some of the CFG rules in Grammar III are replaced by a single ID-rule. For example, the two CFG rules:

$$\texttt{vp=tense} \longrightarrow \texttt{vp=tense advp}$$

$$\texttt{vp=tense} \longrightarrow \texttt{advp vp=tense}$$

(**vp=tense** and **advp** stand for "tensed verb phrase" and "adverbial phrase", respectively) are replaced by a single ID-rule:

$$\texttt{vp=tense} \xrightarrow{\text{ID}} \texttt{advp, vp=tense}$$

The 40 test sentences are also from [77, Appendix G, pp.185-9]. In the following table, the sentences in [77] are referred to by their Sentence No.'s; length refers to the number of words in the sentence; parsing time is measured in seconds.

Sentence No.	Length	Parse Time	Sentence No.	Length	Parse Time	Sentence No.	Length	Parse Time	Sentence No.	Length	Parse Time
1	19	0.213	11	18	0.176	21	19	0.167	31	23	0.205
2	11	0.064	12	19	0.166	22	14	0.059	32	30	0.248
3	26	0.281	13	19	0.139	23	13	0.085	33	28	0.255
4	8	0.037	14	12	0.076	24	7	0.030	34	26	0.278
5	12	0.064	15	11	0.058	25	7	0.034	35	23	0.165
6	13	0.071	16	14	0.135	26	9	0.048	36	1	0.001
7	11	0.079	17	13	0.112	27	13	0.133	37	2	0.001
8	7	0.034	18	17	0.136	28	34	0.279	38	4	0.013
9	15	0.072	19	27	0.147	29	8	0.050	39	5	0.027
10	22	0.147	20	13	0.082	30	28	0.248	40	5	0.043